ioProgrammo

PER ESPERTI E PRINCIPIANTI

Rivista + Le Grandi Guide di ioProgrammo n° 3 a € 14,90 in più

VERSIONE PLUS RIVISTA+LIBRO+CD €9,90

VERSIONE STANDARD RIVISTA+CD €6,90

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • NOVEMBRE 2005 • ANNO IX, N.10 (96)

TEST CON LE MACCHINE VIRTUALI

UTILIZZA "VIRTUAL PC" PER INSTALLARE SISTEMI OPERATIVI DIVERSI SULLO STESSO COMPUTER

COME PORTARE LA VOCE IN RETE E RISPARMIARE SULLA BOLLETTA

VOICE OVER IP

PERSONALIZZA LE APPLICAZIONI USATE DALLA TUA AZIENDA INTEGRANDOLE CON FUNZIONI DI TELEFONIA SU INTERNET

TEORIA I protocolli e gli standard da conoscere per diventare un esperto

TECNICA Le librerie da usare per semplificare la programmazione

PRATICA Il codice JAVA per iniziare a programmare subito

C# ADO.NET SOLUZIONE DI UN CASO DI STUDIO

QUANDO IL DATABASE CONTIENE DI TUTTO

Impara come salvare al suo interno suoni, immagini, filmati, classi ed oggetti

C++ NON SOLO VIDEOGIOCHI

SIMULAZIONE DI ESTERNI

Ricrea in 3D, da codice, un luogo geografico e fallo esplorare con mouse e tastiera

■ VB.NET

MONITOR DI SISTEMA

Tre esempi per raccogliere informazioni con SNMP

■ PHP

INTERNET & WEB SERVICES

XML-RPC, ti spieghiamo come comunicano i blog

■ JAVA

ALGORITMI GENETICI

Scopri come funzionano le tecniche più controverse del momento

VISUAL BASIC

FOTOGRAFIE AD EFFETTO

Ecco come ritoccare le immagini direttamente da una nostra applicazione

ASP 2.0

SITI WEB CON LE MASTER PAGES

Usa gli elementi ripetitivi in tutte le pagine senza doverli ricopiare

JAVA

EFFETTO VETRO SULLE FINESTRE

Applicazioni "alla moda" anche per i tuoi software Java, ecco come!

ROBOT WAR

Crea un jRobot e gettalo in un'arena virtuale insieme a tutti gli altri

DATABASE

RIVOLUZIONE IN ARRIVO

Mai più dati sull'Hard Disk, usiamo la memoria per essere molto più veloci!!!

NETWORKING

UN LETTORE DI NEWSGROUP

Come programmare un NNTP reader personalizzandolo per le proprie esigenze

BONJOUR LA RETE È SERVITA

Il protocollo zeroconf, per servizi che annunciano a tutti i PC disponibili

I TRUCCHI RAPIDI

PORTA SERIALE/USB PILOTIAMOLA DA C++

SUDOKU TI SVELIAMO COME TROVARE LE SOLUZIONI GIUSTE UTILIZZANDO UN ALGORITMO INFALLIBILE!

EDIZIONI MASTER www.edmaster.it

Anno IX - N.ro 10 (96) - Novembre 2005 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Raffaele del Monaco, Fabio Farnesi
Collaboratori: M. Autiero, L. Barbieri, C. Bellucci, L. Buono, F. C. Ferracchiati, A Galeazzi, F. Grimaldi, F. Lippo ,A. Maroccelli, L. Mattei, A. Panella, F. Paparoni, A. Pelleriti, F. Smelzo,A. Trapani, F. Vaccaro, V. Vessia.
Segreteria di Redazione Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale: SET S.r.l.
Coordinamento Tecnico: Piero Mannelli
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831213 - Fax 02 83121330
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90
Offerte valide fino al 31/12/05
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore - zona ASI Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Ottobre 2005

Edizioni Master edita: Computer Bild Italia, Computer Games Gold, Digital Japan Magazine, Digital Music, DVD Magazine, Filmteca in DVD, Giochi e Programmi per il tuo telefonino, Go!OnLine Internet Magazine, Guide di Win Magazine, Guide Strategiche di Win Magazine giochi, Home Entertainment, Horror mania, I Corsi di Win Magazine, I Fantastici CD-Rom, I film di idea web,I Filmissimi in DVD, I Libri di Quale Computer, I Mitici all'Italiana, Idea Web, InDVD, IoProgrammo, Japan Cartoon, La mia Barca, La mia Videoteca, Le Grandi Guide diioProgrammo, Linux Magazine, MPC, Nightmare, Office Magazine, PC Junior, Win Junior, PC VideoGuide, Quale Computer, Softline Software World, Supercar in dvd, Thriller Mania, Win Magazine Giochi, Win Magazine, Le Collection.

Alzate la voce!

Avvicinare le persone comuni all'universo tecnologico, questa è stata l'idea "rivoluzionaria" che ha portato alla diffusione di massa dell'Informatica e alla creazione di un "mondo digitale" che fino a qualche anno fa era limitato a pochi fortunati ricercatori. Così fra i sistemi Microsoft certamente orientati a questa mission, Linux che sempre più tenta di recuperare il gap di "facilità d'uso" che fino a qualche anno fa aveva nei confronti della concorrenza, Apple e infine Internet, oggi possiamo affermare che l'utente comune ha in mano tutti i mezzi per utilizzare l'informatica per innalzare la qualità della propria vita. Questa premessa era d'obbligo in relazione all'articolo sul Voice Over IP che pubblichiamo in questo numero. Se da un lato gli utilizzatori della tecnologia non hanno conoscenza di cosa c'è dietro un software e non hanno dunque la possibilità di migliorarlo e adattarlo alle proprie esigenze, dall'altro un programmatore ha uno sguardo completo su ogni singolo bit portatore di un'innovazione significativa. Il Voice Over IP non è una tecnologia particolarmente difficoltosa da implementare, ci si può accostare ad essa con la curiosità richiesta ad un buon programmatore e riuscire a integrare la tecnica all'interno delle proprie applicazioni con uno sforzo tutto sommato non eccessivo. Questo significa avere in mano un mezzo potentissimo per l'abbattimento dei costi di comunicazione, per migliorare l'efficacia della propria azienda, per adattare il software alle proprie esigenze. Potete creare un call center personalizzato, oppure semplicemente un software per la telefonia su internet, e molto altro ancora. Quello che è importante sapere è che alcune tecnologie che appaiono come rivoluzionarie quando portate alla ribalta dai grandi nomi sono invece alla nostra portata. Se sembrano lontani i tempi in cui un manipolo di "hacker" chiusi in uno scantinato progettavano applicazioni che avrebbero cambiato il futuro dell'informatica e creato un mondo così diverso da quello abituale, sappiate che molto probabilmente non è così. Bisogna ritrovare il gusto di sperimentare, proporre, costruire, e immaginare ancora, per creare le applicazioni che fra qualche anno ci faranno pensare al presente come a un tempo lontano.

Fabio Farnesi
ffarnesi@edmaster.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre \soft\codice\ e \soft\tools\) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

VOICE OVER IP

PERSONALIZZA L'APPLICAZIONE USATA DALLA TUA AZIENDA. AGGIUNGI LE FUNZIONI DI TELEFONIA SU INTERNET E RISPARMIA SUL COSTO DELLA BOLLETTA

 I protocolli e gli standard utilizzati

 Le librerie da usare per semplificare la programmazione

 Il codice per iniziare a programmare subito

pag. 16

QUANDO IL DATABASE CONTIENE DI TUTTO

Impara come salvare al suo interno, immagini suoni, filmati e persino classi ed oggetti

pag. 30

IOPROGRAMMO WEB

Web Services con semplicità . pag. 22
In questo articolo illustreremo il protocollo XML-RPC. Utilissimo per applicazioni che fanno uso della programmazione distribuita, ma meno complicato di Soap. Vedremo, infine, un esempio d'uso in PHP

ASP.NET 2.0 Le master Pages . pag. 26
Alla scoperta di una delle novità più interessanti contenute nella nuova versione del Framework di Microsoft. Una tecnica che ci consente di risparmiare tempo e guadagnare in flessibilità

DATABASE

DB Prevalence, la nuova frontiera . . . . pag. 36
Muoviamo i primi passi nella tecnica che tutti i futuri DB stanno sperimentando, è più veloce, più stabile e più comoda da gestire. Cosa ci dovremo aspettare in futuro dai Database?

NETWORKING

NewsReader? Facciamolo con C#! . . . . pag. 42
I newsgroup sono una delle maggiori risorse che la rete offre al suo vasto pubblico. Impariamo come funziona il protocollo per inviare e ricevere messaggi e creiamo un lettore di news

SNMP ti dà il controllo totale . pag. 48
Ecco il protocollo che consente di analizzare il comportamento di qualunque periferica hardware. Collegata a un computer, ad una rete o isolata, non fa differenza, noi la controlliamo!

Bonjour, la rete è fatta! . . . . . pag. 54
Alla scoperta di "Bonjour" una libreria prodotta da Apple che ci consente di creare reti "Autoconfiguranti". In questo appuntamento creeremo un servizio di File Sharing molto particolare...

SISTEMA

Effetto vetro per le finestre. . pag. 60
Inpariamo come accedere alle API di Windows e sfruttare l'accesso al sistema operativo per creare applicazioni dotate di finestre trasparenti. Un effetto decisamente insolito per Java

GAMING

Terrain Mapping gioco o realtà?. . . . . pag. 64
Impariamo a realizzare ambienti virtuali che simulano gli spazi aperti. Fra boschi, montagne, terreni sconfinati, utilizzeremo il computer per realizzare ogni nostra fantasia!

GAMING

Guerra dei Robot Chi vincerà? pag. 68
Creiamo in Java un automa e iscriviamolo a un moderno torneo a squadre pronto a sfidare migliaia di altri robot. Chi sarà il migliore?

VISUAL BASIC

Effetti speciali sulle immagini. pag. 72
In questo articolo vedremo come trasformare un'immagine, in base ai valori di una funzione, per ottenere fantastici effetti speciali

BACKSTAGE

Programmare con Darwin . . . pag. 78
Se ne parla ormai da moltissimo tempo, ma cosa sono e come funzionano gli algoritimi genetici? Come possiamo usarli, in modo intelligente, per sviluppare software che migliora nel tempo?

CORSI

XSL • Il trasformista per eccellenza..........................pag. 88
Una tecnica comoda, capace di trasformare qualunque docuimento XML in un altro formato. Facile da usare e utilizzabile da qualunque linguaggio. Scopriamola insieme!

VB.NET • La Gestione degli Errori...............................pag. 94
In questo appuntamento impareremo come costruire applicazioni a prova di errore! Gestiremo tutte le possibili combinazioni nel tentativo di produrre software sempre più affidabile

SOFTWARE

Macchine Virtuali Soluzione per i test! . . . . pag. 100
Utilizziamo VirtualPC per installare su una sola macchina fisica molti sistemi operativi. Lo scopo è simulare il comportamento del software in condizioni diverse

SOLUZIONI

Programmare sudoku. . . . . . pag. 109
Il passatempo più in voga del momento arriva dal Giappone e si chiama sudoku. La programmazione svela alcuni interessanti aspetti del gioco e ci intriga quanto, o addirittura di più, del gioco stesso delle nostre applicazioni in ambienti eterogenei

http://forum.ioprogrammo.it

RUBRICHE

Gli allegati di ioProgrammo pag. 6
Il software in allegato alla rivista

News pag. 10
Le più importanti novità del mondo della programmazione

La posta dei lettori pag. 12
L'esperto risponde ai vostri quesiti

Tips & Tricks pag. 84
Trucchi per risolvere i problemi più comuni

Express pag. 86
Le guide passo passo per realizzare applicazioni senza problemi

Software pag. 104
I contenuti del CD allegato ad ioProgrammo. Corredati spesso di tutorial e guida all'uso

Biblioteca pag. 114
I migliori testi scelti ogni mese dalla redazione per aiutarvi nella programmazione

QUALCHE CONSIGLIO UTILE

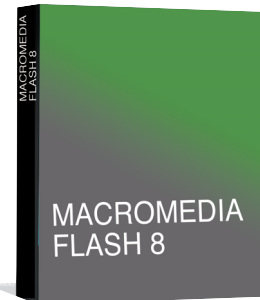
I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

LE NUOVE VERSIONI DEI TOOL PIÙ ATTESI PER IL WEB E IL MULTIMEDIA

MACROMEDIA FLASH 8

Grafica, immagini, suoni, animazioni, codice, tutto condensato in un unico prodotto!

MACROMEDIA DREAMWEAVER 8

L'IDE evoluto che ti consente di sviluppare siti WEB in modo rapido ed intuitivo

RIVISTA + CD-ROM in edicola

Prodotti del mese

Phalanger 1.0 RC1

NOVITÀ ECCEZIONALE

L'ADD-IN che fa girare le applicazioni PHP compilandole in NET

Il unzionamento di Phalanger è banale da un punto di vista fisico, più complesso da un punto di vista logico. Da un lato consente di sviluppare applicazioni PHP tramite Visual Studio.NET. Queste applicazioni possono essere ricompilate sotto forma di eseguibile. Dal punto di vista del web Phalanger non ha bisogno del classico interprete PHP per funzionare, viceversa utilizza un modulo .NET che ricompila gli script e affida l'esecuzione al .NET Framework. In pratica si tratta di PHP per .NET. Esattamente come esiste C#,VB.NET con cui potete creare le vostre pagine .ASPX adesso potete immaginare di usare PHP .NET con tutti i vantaggi della tecnologia in questione in ambienti Microsoft. Il processo di installazione è molto semplice, tuttavia per motivi di licenza vi cerrà chiesto di scaricare dei prerequites dal web.

[pag.106]

MySQL Migration Toolkit

Il software per migrare da Oracle, Access, SQL Server a MySQL senza problemi!

A quanto pare MySQL vuole fare le cose in grande, così ecco arrivare un wizard completamente grafico per migrare i dati da db come Oracle e Access a MySQL. Si tratta di una bella comodità e di un invito esplicito a provare MySQL anche in ambiti dove fino ad ora non aveva trovato terreno fertile.
Se è logico infatti migrare da Access a MySQL sulla base di considerazioni legate alla multiutenza, alla velocità e a tutti i vantaggi di questa migrazione, bisogna essere sicuri di offrire un prodotto competitivo per chiedere di migrare da un mostro sacro come Oracle a MySQL.
In ogni caso questo wizard rappresenta la soluzione ottimale per effettuare la migrazione senza troppi problemi di conversione dati o di perdita di informazioni.

[pag.107]

Eclipse SDK 3.1

L'ambiente universale e personalizzabile per sviluppare con qualsiasi linguaggio!

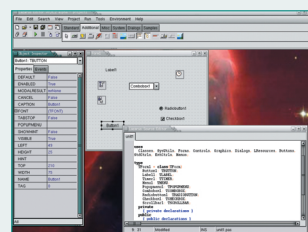
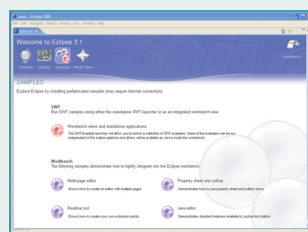
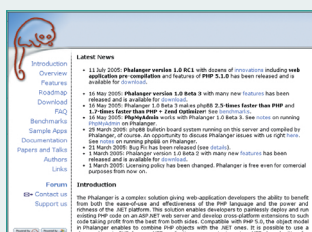
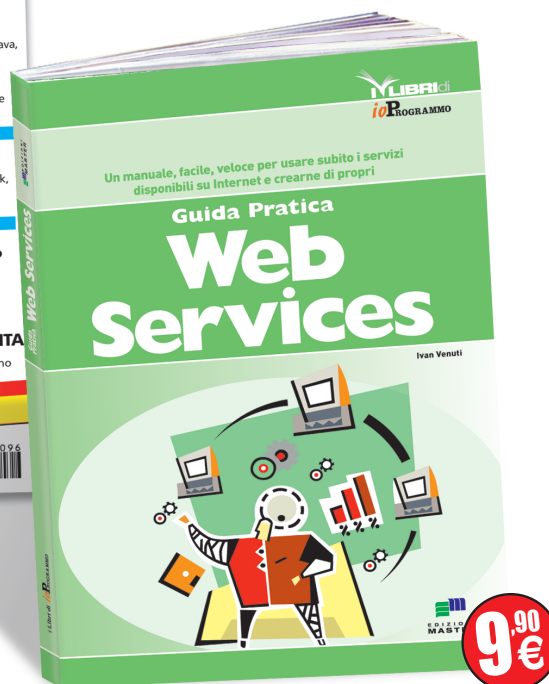
Eclipse nasce da un'intuizione straordinaria, ovvero quello di essere un ambiente estensibile per la programmazione. Nato come ambiente preferito per la programmazione Java, si è evoluto diventando decisamente "multilinguaggio". Installando il plugin giusto vi troverete sotto mano sempre un IDE aggiornato per lo sviluppo con il vostro linguaggio preferito. Inoltre tutte le funzionalità necessaria ad un buon programmatore per sviluppare rapidamente il proprio codice sono garantite dalla base comune dell'ambiente, il che rende Eclipse uno strumento assolutamente straordinario.
Aggiungete tutto ciò che si tratta di un IDE gratuito e multipiattaforma e scoprirete perché rapidamente è diventato il preferito dai programmatori.

[pag.105]

Lazarus

Il clone gratuito di Delphi, per sviluppare in modo rapido, visuale e gratuito

I programmatori più anziani ricorderanno Delphi come il primo ambiente realmente RAD ad essere stato immesso sul mercato. È grazie a Delphi se oggi ragioniamo in termini di componenti, ed è grazie a Delphi che è nato il concetto di Rapid Application Development. Non mancherebbe davvero niente a questo ambiente per farne il leader indiscusso del mercato, se non fosse che un prezzo particolarmente elevato e qualche strategia non troppo azzeccata lo hanno relegato ad essere uno degli ambienti preferiti in applicazioni di tipo Business, ma non il più diffuso in ambienti Home/Office. A modificare la situazione ci pensa Lazarus che, non contiene tutte le soluzioni proposte da Delphi, ma ha dalla sua il fatto di essere completamente Free e di rappresentare un clone praticamente perfetto della versione base di Delphi

[pag.106]

I contenuti del libro

GUIDA PRATICA Web Services

Un manuale pratico, veloce, di rapida consultazione per conoscere i servizi da utilizzare subito e realizzarne di propri. Ivan Venuti ci proietta in un universo fatto di connessioni, di applicazioni capaci di utilizzare parti di codice sparse su Internet indipendentemente dal linguaggio e dalla piattaforma.
Non è solo un manuale questo, è piuttosto un'indicazione che vi proietta nella programmazione del futuro, la dove il lavoro del singolo si fonde con quello di milioni di altre persone, il tutto per creare l'applicazione distribuita più grande di tutti i tempi.
Un viaggio affiscinante, percorso in modo pratico, utilizzando gli strumenti della tecnica e pronto per essere utilizzato subito.

- **Architetture Distribuite**
- **Il primo Web Services in...un Battibaleno**
- **Tecnologie Sottostanti i Web Services**
- **Collaborare? WS-I Basic Profile**
- **Realizzare un WSDL Partendo da Zero**
- **Axis in Dettaglio**
- **Web Services in Java? Non solo Axis**
- **Realizzare un Client nei Diversi Linguaggi**
- **Client e Servizi con Messaggi Soap Wrapped/Lieral**

News

SUPERPREMIO PER CHI COSTRUIRÀ L'EMULATORE DI WINDOWS XP

SpecOps Labs già nota per il tentativo di costruzione di “David” l'emulatore del sistema operativo Windows ha messo in palio ben 10,000 dollari per il programmatore che riuscirà a produrre un software che consenta di eseguire e installare applicazioni compatibili con XP sul sistema operativo Linux.

INTERNET EXPLORER, NUOVE FALLE

È una notizia ormai di quelle che risultano così popolari da non destare stupore, ma è una notizia comunque da dare per il numero di utenti che essa coinvolge. Così un'altra falla sarebbe stata scoperta in IE, e quel che più conta si tratta di una falla che si applica anche a sistemi dotati di Windows XP attrezzato con il service pack 2. Non si tratta questa volta di un worm ma di una falla che consentirebbe ad utenti maliziosi di eseguire codice remoto sulla macchina client. Al momento in cui scriviamo non è stata rilasciata ancora alcuna patch ufficiale nè sono stati diffusi i particolari sul bug in questione.

ARRIVA IL PRIMO VIRUS CROSS PLATFORM

Trend Micro avverte: è in arrivo Cardtrp, un virus piuttosto insidioso in grado di infettare allo stesso modo sia i cellulari basati sulla piattaforma Series 60 che i pc con sistema operativo Windows. Il meccanismo di propagazione del Worm sembra essere piuttosto ingegnoso. Il virus infatti deposita nella memory card del telefono un file autorun.inf. Nel momento in cui la memory card viene letta da un card reader di un PC, il file va in autoesecuzione e installa sul computer vittima due virus: il Wukill.B e il Bkdr_Berbew.A.

UNA TRILOGIA IN ARRIVO DALLA PROFESSIONAL DEVELOPER CONFERENCE

Si è tenuta nel mese di Settembre la tradizionale Professional Developer Conference. La PDC è uno dei maggiori eventi annuali che coinvolgono gli sviluppatori. Organizzata da Microsoft, in Los Angeles, dal 13 al 16 Settembre 2005 ha visto la partecipazione di migliaia di programmatori tutti in attesa di conoscere le novità che Microsoft proporrà al mondo dello sviluppo nel prossimo anno. Lo Show è stato assolutamente all'altezza delle aspettative, ma al di la dell'aspetto organizzativo, a sorprendere è stata sicuramente la

CODICE OPENSOURCE IN WINDOWS SERVER 2003

Per ora si tratta solo di indiscrezioni, ma Kyril Faenov, direttore per l'High Performance Computing in Microsoft, avrebbe dichiarato in una recente intervista che il suo gruppo di lavoro sta sperimentando l'uso di un componente Open Source da includere in Windows Server 2003 Compute Cluster Edition.
Scavando un po' più a fondo sembrerebbe che la libreria in questione sia la "Message Passing Interface".
La MPI è un middleware sviluppato nel 1990 da un consorzio di aziende, la cui funzione è quella di proporre uno standard per lo scambio di messaggi, che favorisca la portabilità del codice. Non si tratta dunque di un componente vitale del sistema. È comunque interessante notare come la validità di un progetto non sia collegata alla sua licenza ma semplicemente alla sua utilità.
Microsoft avrebbe comunque ottimizzato la libreria, e l'avrebbe rinominata MPICH2. Il poter partire da una base solida come quella di MPI sicuramente a fatto risparmiare a MS un bel po' di energie. MPI è distribuita sotto licenza BSD e non sotto GPL, ci chiediamo come MS restituirà la cortesia alla comunità OpenSource

densità dei contenuti presentati. Microsoft ha messo in mostra un "mondo" di iniziative per il prossimo anno, ma sempre e comunque ogni cosa ruoterà intorno a due paletti fondamentali: "Windows Vista" e "Office 12". Attorno a questi due elementi vivranno una serie di tool e tecnologie indispensabili per noi programmatori.
Una delle novità più interessanti è stata la presentazione di un trio molto particolare, costituito da "Acrylic Graphic Design, Sparkle Interactive Design e Quartz Web Designer". I nomi sono evocativi della volontà di Microsoft di trasformare i programmatori da editori di codice a designer di applicazioni a tutto tondo. I tre tool in questione serviranno ai programmatori a disegnare le applicazioni prima che a svilupparle. E se da un lato si è fatto e detto di tutto per affermare che non si tratta di un modo per andare in competizione con i software di Adobe/Macromedia, appare evidente come questo scontro sarà probabilmente inevitabile.
Staremo a vedere.

UNO SCANNER/OCR NEL CELLULARE

Presto una nuova funzionalità potrebbe aggiungersi ai nostri già ipertecnologici cellulari. Semplicemente "passando" il cellulare sulla superficie di una pagina potremo infatti effettuarne la scansione. La tecnica è stata sviluppata da NEC in collaborazione con il Naist - Nara Institute of Science and Technology - con sede, ovviamente in Giappone. A differenza di alcuni cellulari attualmente in commercio che incorporano funzionalità di un rudimentale OCR, i nuovi telefoni consentiranno di scansionare immagini di dimensioni considerevoli. L'immagine originale verrà scomposta in dozzine di immagini più piccole proporzionate all'ottica dell'obiettivo contenuto nel telefono. Sarà poi il software a ricostruire il testo originale, a correggere eventuali difetti nella curvatura della pagina ed a restituire un'immagine scansionata di alta qualità. Alcuni test hanno dimostrato come con una camera da un megapixel, si impiegano circa 5 secondi a scansionare un documento in formato A4, e l'immagine originale viene ricomposta da un sottoinsieme compreso fra le 21 e le 35 pagine. Presto dunque il nostro cellulare diventerà anche una fotocopiatrice, e ovviamente ci sarà di nuovo qualche polemica sul copyright.
L'idea è sempre la stessa, non sono gli strumenti tecnologici ad essere dannosi, ma il loro uso distorto sicuramente potrebbe esserlo.

BRUXELLS DISCUTE SULL'ISTITUTO EUROPEO PER LA TECNOLOGIA

La commissione europea, dibatte in questo periodo, sull'opportunità di creare un'Istituto Europeo per la tecnologia". L'idea è quella di creare un centro di raccolta per le migliori menti informatiche /tecnologiche del vecchio continente, che possa fare da volano all'economia basata sullo sviluppo tecnologico. Il presidente Barroso più volte si è espresso favorevolmente rispetto all'opportunità della creazione di una rete di esperti collocata al centro dell'Europa. Al di la delle opportunità di sviluppo e di conseguenza dell'innalzamento del livello occupazionale appare chiaro come l'esistenza dell'Istituto potrebbe arginare quella fuga di menti che affligge da più tempo il vecchio continente. In mancanza di strutture affidabili, scienziati, tecnici, ed esperti preferiscono infatti migrare verso Stati uniti e Asia.

C# 3.0 PRIME INDISCREZIONI

Anders Hejlsberg è un mito per qualunque programmatore. È lui il padre del Turbo Pascal prima, di J++ dopo e infine anche di C#. Si tratta di uno di quei personaggi che in un qualche modo hanno influenzato in maniera profonda la storia dell'informatica. Recentemente questo mostro sacro della programmazione ha rilasciato qualche dichiarazione in relazione alle novità che secondo lui rappresenteranno il punto di svolta più interessante per il prossimo C# 3.0. In particolare Anders Hejlsberg ha focalizzato l'attenzione su LINQ. Il Language INtegrated Query è stato recentemente presentato anche nel corso della professional developer conference. Si tratta di un linguaggio che affianca il tradizionale SQL e consente al programmatore di effettuare delle ricerche all'interno di strutture dati direttamente in modo nativo. Ad esempio disponendo

di un array di stringhe, sarà possibile utilizzare una query LINQ per ottenerne un sottoinsieme basato su un filtro di ricerca.
Allo stesso modo disponendo di un set di dati recuperati da un DataBase sarà possibile interrogarlo tramite LINQ. La novità è evidente, LINQ è un un'interfaccia di interrogazione universale per qualunque tipo di dato, sia esso un database o un array di stringhe o una qualsiasi altra fonte.

densità dei contenuti presentati. Microsoft ha messo in mostra un "mondo" di iniziative per il prossimo anno, ma sempre e comunque ogni cosa ruoterà intorno a due paletti fondamentali: "Windows Vista" e "Office 12". Attorno a questi due elementi vivranno una serie di tool e tecnologie indispensabili per noi programmatori.
Una delle novità più interessanti è stata la presentazione di un trio molto particolare, costituito da "Acrylic Graphic Design, Sparkle Interactive Design e Quartz Web Designer". I nomi sono evocativi della volontà di Microsoft di trasformare i programmatori da editori di codice a designer di applicazioni a tutto tondo. I tre tool in questione serviranno ai programmatori a disegnare le applicazioni prima che a svilupparle. E se da un lato si è fatto e detto di tutto per affermare che non si tratta di un modo per andare in competizione con i software di Adobe/Macromedia, appare evidente come questo scontro sarà probabilmente inevitabile.
Staremo a vedere.

UNO SCANNER/OCR NEL CELLULARE

Presto una nuova funzionalità potrebbe aggiungersi ai nostri già ipertecnologici cellulari. Semplicemente "passando" il cellulare sulla superficie di una pagina potremo infatti effettuarne la scansione. La tecnica è stata sviluppata da NEC in collaborazione con il Naist - Nara Institute of Science and Technology - con sede, ovviamente in Giappone. A differenza di alcuni cellulari attualmente in commercio che incorporano funzionalità di un rudimentale OCR, i nuovi telefoni consentiranno di scansionare immagini di dimensioni considerevoli. L'immagine originale verrà scomposta in dozzine di immagini più piccole proporzionate all'ottica dell'obiettivo contenuto nel telefono. Sarà poi il software a ricostruire il testo originale, a correggere eventuali difetti nella curvatura della pagina ed a restituire un'immagine scansionata di alta qualità. Alcuni test hanno dimostrato come con una camera da un megapixel, si impiegano circa 5 secondi a scansionare un documento in formato A4, e l'immagine originale viene ricomposta da un sottoinsieme compreso fra le 21 e le 35 pagine. Presto dunque il nostro cellulare diventerà anche una fotocopiatrice, e ovviamente ci sarà di nuovo qualche polemica sul copyright.
L'idea è sempre la stessa, non sono gli strumenti tecnologici ad essere dannosi, ma il loro uso distorto sicuramente potrebbe esserlo.

BRUXELLES DISCUTE SULL'ISTITUTO EUROPEO PER LA TECNOLOGIA

La commissione europea, dibatte in questo periodo, sull'opportunità di creare un'Istituto Europeo per la tecnologia". L'idea è quella di creare un centro di raccolta per le migliori menti informatiche /tecnologiche del vecchio continente, che possa fare da volano all'economia basata sullo sviluppo tecnologico. Il presidente Barroso più volte si è espresso favorevolmente rispetto all'opportunità della creazione di una rete di esperti collocata al centro dell'Europa. Al di la delle opportunità di sviluppo e di conseguenza dell'innalzamento del livello occupazionale appare chiaro come l'esistenza dell'Istituto potrebbe arginare quella fuga di menti che affligge da più tempo il vecchio continente. In mancanza di strutture affidabili, scienziati, tecnici, ed esperti preferiscono infatti migrare verso Stati uniti e Asia.

C# 3.0 PRIME INDISCREZIONI

Anders Hejlsberg è un mito per qualunque programmatore. È lui il padre del Turbo Pascal prima, di J++ dopo e infine anche di C#. Si tratta di uno di quei personaggi che in un qualche modo hanno influenzato in maniera profonda la storia dell'informatica. Recentemente questo mostro sacro della programmazione ha rilasciato qualche dichiarazione in relazione alle novità che secondo lui rappresenteranno il punto di svolta più interessante per il prossimo C# 3.0. In particolare Anders Hejlsberg ha focalizzato l'attenzione su LINQ. Il Language INtegrated Query è stato recentemente presentato anche nel corso della professional developer conference. Si tratta di un linguaggio che affianca il tradizionale SQL e consente al programmatore di effettuare delle ricerche all'interno di strutture dati direttamente in modo nativo. Ad esempio disponendo

di un array di stringhe, sarà possibile utilizzare una query LINQ per ottenerne un sottoinsieme basato su un filtro di ricerca.
Allo stesso modo disponendo di un set di dati recuperati da un DataBase se sarà possibile interrogarlo tramite LINQ. La novità è evidente, LINQ è un un'interfaccia di interrogazione universale per qualunque tipo di dato, sia esso un database o un array di stringhe o una qualsiasi altra fonte.

Che cosa vuol dire WEP?

Gentile redazione, da un po' di tempo sento parlare di sicurezza nelle reti wireless. Sono a conoscenza del fatto che in America, ma anche in Italia ormai il tentativo di violare la riservatezza delle reti wireless è piuttosto diffuso. Mi pare di avere capito che il miglior metodo per proteggere la propria rete è quello di adottare una protezione WEP. Ma non ho ben chiaro il perché questo standard dovrebbe consentirmi di restare al riparo da eventuali attacchi.

Marco da Salerno

Gentile Marco, WEP è l'acronimo per Wired Equivalent Privacy, si tratta del meccanismo di protezione adottato internamente dallo standard 802.11 che è utilizzato nella trasmissione dei pacchetti via onde radio. C'è da dire che il WEP non è un algoritmo di autenticazione, piuttosto è un algoritmo di crittazione dei dati. Perciò in una architettura Wireless sono da prevedere due fasi, una che identifichi l'utente e gli consenta di accedere alla rete wireless e una che consenta di trasmettere i dati in maniera criptata e sicura. Per quanto riguarda la fase di autenticazione, generalmente vengono utilizzate due modalità: OSA e SKA. Nella modalità OSA - Open Systems Authentication lo standard non prevede nessun processo di autenticazione e l'accesso è garantito a chiunque voglia usufruire della rete. Nella modalità SKA - Shared Key Authentication viceversa, quando l'Access Point riceve una richiesta d'accesso da parte di un terminale mobile, gli invia una chiave generata un modo del tutto casuale. Il terminale firma la chiave utilizzando a sua volta una chiave "Pre-Condivisa" e restituisce il dato all'Access point. Quando l'AP riceve il dato lo confronta utilizzando la chiave random più la chiave "pre-condivisa" in suo possesso, se i due dati coincidono concede l'autenticazione. Una volta autenticato l'utente è in grado di trasmettere/ricevere dati in modo criptato. Il WEP si basa su un algoritmo denominato RC4. In questo genere di algoritmo sono coinvolti un vettore di inizializzazione IV a 24 bit, e una chiave condivisa a 40 bit. A questo punto l'algoritmo diventa piuttosto complesso. La chiave Wep di 40 bit viene concatenata al vettore di inizializzazione, questa unione genera una stringa 64 bit che viene data in pasto a RC4 che la usa come chiave per la cifratura dei dati. Al di la di come poi vengano realmente trasmessi e cifrati i dati, è importante a questo punto sottolineare il ruolo del vettore di inizializzazione. Il vettore in questione viene utilizzato per rendere il flusso criptato sempre diverso ad ogni trasmissione. Tuttavia è proprio il vettore di inizializzazione il punto debole della cifratura WEP. La lunghezza dell'IV infatti è di 24 bit, pertanto ammette solo 2^{24} il vettore di inizializzazione essendo soltanto lungo 24 bit, ammette uno spazio di solo 2^{24} combinazioni. Questo vuol dire che in un paio d'ore di intercettazioni è abbastanza facile riprodurre la chiave WEP. Per risolvere questo problema è nato WPA che supporta una chiave a 128 bit e un vettore di inizializzazione a 48 bit.

Le novità di XAML

Cari guru di ioProgrammo, ad ogni click di browser sono sommerso di informazioni su Windows Vista, Avalon, WinFS etc. Inutile dire che la mia confusione è alle stelle. Ad incuriosirmi maggiormente è la criptica sigla XAML , che credo abbia qualcosa a che fare con la grafica.
Mi spiegate esattamente di cosa si tratta?

Enrico da Modena

Caro Enrico. Windows Vista è ormai alle porte, senza contare che il clamore suscitato dalle demo viste alla recente Professional Developer Conference aumenta sensibilmente i rumors che riguardano il prossimo sistema operativo di Microsoft. Semplificando abbastanza possiamo dire che Avalon è la piattaforma su cui si baserà l'intero sistema grafico di Windows Vista. A sua volta Avalon sfrutta abbondantemente XML per la creazione di bottoni, pulsanti e gli altri elementi grafici che caratterizzano le interfacce. In particolare il linguaggio XML sfruttato da Avalon prende il nome di XAML. Da un lato si può dire che XAML è un linguaggio per la creazione di elementi grafici, dall'altro si può dire che XAML è un linguaggio per la rappresentazione di oggetti. Ogni tag esistente in un file di definizione XAML viene convertito infatti in corrispondenti Oggetti in fase di esecuzione, il chè significa che in una certa qual misura potrebbe essere considerato come un ponte verso il vecchio WinForms.
Lo spazio non ci concede di andare ulteriormente in profondità ma sicuramente avremo modo di parlare ancora sia di Avalon che di XAML e ovviamente di tutto quello che circonda il nuovo Windows Vista.

Creare utenti in mysql

Salve, ho apprezzato molto il libro "Imparare SQL" che avete allegato in un recente numero di ioProgrammo. Sto cercando di fare i miei primi esperimenti, vorrei capire come accedere a mysql per creare un nuovo utente, mi date una mano?

Antonio da Rimini

Salve Antonio, l'operazione per la creazione di un nuovo utente è semplice. Prima di tutto devi avere installato mySQL e che il servizio sia avviato. Se tutto è a posto, apri una console dos, oppure se sei sotto Linux un terminale e digita:

```
mysql -uroot -ppassword
```

Dove password è la password dell'utente root. Questo ti farà accedere all'amministrazione del server. Per creare un nuovo utente puoi usare:

```
GRANT ALL PRIVILEGES ON *.* TO
  'user'@'localhost' IDENTIFIED BY 'some_pass'
                          WITH GRANT OPTION;
```

Questo comando che crea un utente che ha accesso a tutti i db e tutte le tabelle di ogni tabella. Se vuoi puoi limitare gli accessi con qualcosa del genere

```
GRANT SELECT,INSERT,UPDATE,DELETE,
                          CREATE,DROP
-> ON bankaccount.*
-> TO 'custom'@'localhost'
-> IDENTIFIED BY 'password';
```

che da l'accesso per l'utente *custom @localhost* solo su tutte le tabelle del db bankaccount e solo con i permessi citati.
Attenzione ancora che in mysql *custom @localhost* è diverso da *custom @192.168 .x.x*

Software creato da dipendente

Cara redazione di ioProgrammo, ho trascorso diversi anni a lavorare come dipendente sempre per la stessa azienda, per la quale ho sviluppato parecchio software atto a fare funzionare meglio l'attività. Dopo molti anni di lavoro mi è stato dato il benservito. Ho richiesto che il software da me prodotto non fosse più utilizzato dall'azienda, perché la ritengo una mia opera d'ingegno e pertanto non appartiene al corredo aziendale. Ovviamente mi è stato negato. Cosa posso fare?

Emanuele da Genova

Emanuele, prima di tutto sarebbe opportuno consultare un esperto legale, illustrandogli in ogni dettaglio la tua situazione. In linea del tutto generale, possiamo dirti che a meno di accordi preesistenti: "i programmi creati dal lavoratore dipendente nell'esecuzione delle sue mansioni i diritti di utilizzazione economica spettano al datore di lavoro". Pertanto se la situazione è esattamente quella che ci hai illustrato dubitiamo che tu possa vantare dei diritti sulla tua opera.

Registrare un software

Gentile redazione, ho appena creato un software dedicato alla gestione di studi commerciali. poiché intendo vendere la mia opera, vorrei capire se e quanto registrare il software presso la SIAE per tutelare i miei diritti. Inoltre vorrei avere informazioni sulle modalità da seguire per effettuare la registrazione.

Gino da Livorno

Gentile Gino, prima di tutto è opportuno dire che esiste un "Registro pubblico per il software", in funzione presso la Sezione OLAF della Direzione Generale della SIAE a cui è stato affidato tramite il Decreto Legislativo 29/12/1992, n. 518. In questo registro vengono contenute opere "che hanno carattere creativo, inteso come carattere di originalità rispetto ai software preesistenti".
La persona riconosciuta come Autore di un software presente nel suddetto registro gode per 70 anni dei diritti morali sulla creazione dell'opera e sui diritti economici derivanti dalla sua commercializzazione. La procedura per la registrazione del software non è complessa: l'autore deve inviare alla SIAE una "dichiarazione", "una descrizione" e una copia del software riprodotta su CDROM. Il modulo da utilizzare per fornire i dati richiesti è il 349 reperibile all'indirizzo *http://www .siae.it/Olaf_sw.asp?click_level=1200.0300.0300&link_page=Olaf_Sw_ModalitaSoftware.htm#doc,* tale modulo deve essere spedito alla SIAE corredato di una marca da bollo da 14,62 €.
A tutto questo va aggiunto il pagamento di un diritto fisso sempre da versare alla SIAE di euro 107,24 per ciascun programma da registrare.
In cambio di tutto ciò la SIAE rilascerà un attestato di avvenuta registrazione nella quale vengono riportati anche la data di registrazione e un numero identificativo univoco. Fatto questo sarai il felice possessore dei diritti sull'opera che tu hai creato.

A chi spedire la posta?

Alla nostra redazione arriva spesso un considerevole numero di email riguardante temi specifici. Per consentirci di risponderevelocemente e in modo adeguato alle vostre domande abbiamo elaborato una FAQ – Frequently Ask Question – o risposte alle domande frequenti in italiano, di modo che possiate indirizzare le vostre richieste in modo mirato.

Problemi sugli abbonamenti

Se la tua domanda ha a che fare con una delle seguenti:

- Vorrei abbonarmi alla rivista, che devo fare?
- Sono un abbonato e non ho ricevuto la rivista, a chi devo rivolgermi?
- Sono abbonato ma la posta non mi consegna regolarmente la rivista, a chi devo rivolgermi?

Contatta **abbonamenti@edmaster.it** specificando che sei interessato a ioProgrammo. Lascia il tuo indirizzo email e indica il numero dal quale vorresti far partire l'abbonamento. Verrai contattato al più presto. Oppure puoi chiamare lo **02 831212**

Problemi sugli allegati

Se riscontri un problema del tipo:

- Ho acquistato ioProgrammo ed il Cdrom al suo interno non funziona. Chi me lo sostituisce?
- Ho acquistato ioProgrammo ma non ho trovato il cd/dvd all'interno, come posso ottenerlo?
- Vorrei avere alcuni arretrati di ioProgrammo come faccio?

Contatta **servizioclienti@edmaster.it**
Non dimenticare di specificare il numero di copertina di ioProgrammo e la versione: con libro o senza libro. Oppure telefona allo **02 831212**

Assistenza tecnica

Se il tuo problema è un problema di programmazione del tipo:

- Come faccio a mandare una mail da PHP?
- Come si instanzia una variabile in c++?
- Come faccio a creare una pagina ASP.NET

o un qualunque altro tipo di problema relativo a tecniche di programmazione, esplicita la tua domanda sul nostro forum: **http://forum.ioprogrammo.it**, uno dei nostri esperti ti risponderà. Le domande più interessanti saranno anche pubblicate in questa rubrica.

Problemi sul codice all'interno del CD

Se la tua domanda è la seguente

- Non ho trovato il codice relativo all'articolo all'interno del cd

Consulta la nostra sezione download all'indirizzo **http://cdrom .ioprogrammo.it**, nei rari casi in cui il codice collegato ad un articolo non sia presente nel cdrom, senza dubbio verrà reso disponibile sul nostro sito.

Idee e suggerimenti

Se sei un programmatore esperto e vuoi proporti come articolista per ioProgrammo, oppure se hai suggerimenti su come migliorare la rivista, se vuoi inviarci un trucco suggerendolo per la rubrica tips & tricks invia una email a **ioprogrammo@edmaster.it**

VoIP, telefonia su internet

Prima parte

VoIP sta assumendo un ruolo importante in internet e Java dispone di librerie per sviluppare applicazioni che facciano sentire la propria voce nella rete. Vediamo come funzionano

L'idea è molto semplice: utilizzare internet come mezzo di trasmissione della voce, per soppiantare gli antiquati telefoni basati su doppino telefonico. Il concetto è sempre lo stesso, convogliare la voce in pacchetti digitali compressi, e farli decomprimere da un software/hardware appropriato al momento della ricezione. Le motivazioni sono ovvie: abbattimento dei costi, migliore qualità del digitale. Le problematiche sono note, la banda non è ancora sufficientemente grande da supportare una tale quantità di dati, è necessario possedere hardware dedicato se si vogliono raggiungere risultati soddisfacenti. Nonostante questo, i vari Skype e Google Talk fanno da apripista ad una tipologia di applicazioni che muoverà l'economia su larga scala. Perciò è il caso di imparare come dotare le nostre applicazioni della possibilità di interazione vocale.

VOICE OVER INTERNET PROTOCOL

VoIP, è il nome con il quale si vuole realizzare la telefonia per mezzo di internet. Gli obiettivi da raggiungere sono diversi. Tra essi, si possono sottolineare 3 aspetti in particolare: l'instradamento di una telefonata per mezzo del protocollo IP, la gestione di eventi Real Time e la Qualità del Servizio.
L'instradamento di una telefonata VoIP tra due computer avviene, fisicamente, per mezzo di internet utilizzando il protocollo IP e scambiando pacchetti che contengono "voce". Questo principio può essere esteso anche a telefoni compatibili, sia fissi che cellulari, che possono accedere a VoIP per mezzo di opportuni gateway.
Naturalmente per collegare a livello software due computer serve un protocollo e SIP, *Session Initiation Protocol,* è lo standard che si sta affermando per stabilire, modificare e terminare sessioni multimediali. SIP è un protocollo che si completa con altri, come RTP, *Real-Time Transport Protocol,* per fornire una Qualità del Servizio adeguata.
Considerando i diversi dispositivi che compongono la struttura di una chiamata su internet, Java ha sviluppato quattro tipologie di api: *Jain-Sip* e *Jain-Sip Lite* API rispettivamente a basso e ad alto livello per applicazioni che usano la J2SE, Sip-Servlet per sfruttare J2EE e Sip per J2ME per i cellulari. Inoltre c'è RTP e per questo Java propone le API di JMF, *Java Media Framework*. Per capire l'utilizzo delle API di Java è utile entrare nei dettagli del protocollo sul quale si regge VoIP.

SIP

SIP è un protocollo basato su testo, come HTTP e SMTP, e scambia messaggi di richiesta e risposta tra applicazioni formati da header e body. I suoi punti di forza sono la semplicità, l'estendibilità e la sicurezza. I servizi

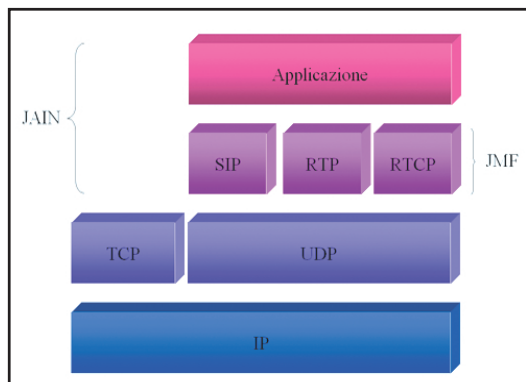



Fig. 1: **Protocolli e API per il VoIP**

che offre vanno dalla localizzazione, caratteristiche e disponibilità dell'utente all'inizializzazione e gestione della chiamata.
I componenti che formano una rete SIP sono cinque: *User Agent Client* (UAC), *User Agent Server* (UAS), *Proxy Server, Redirect Server* e *Registrar Server.* UAC è l'utente che vuole effettuare una telefonata e per farlo invia uno dei possibili messaggi ad un UAS che accetta la richiesta e risponde di conseguenza. Nella richiesta si specifica indirizzo IP e porta del UAS o il suo URI. UAS è dunque il server che riceve la richiesta da uno UAC. Questo modo diretto di comunicare tra UAC e UAS è un tipico esempio PTP, Peer to Peer cioè comunicazione punto punto tra due computer. Naturalmente, come avviene per questo genere di comunicazioni, i ruoli si invertono continuamente e chi manda il messaggio è sempre considerato client e chi lo riceve server; nel momento in cui il server vuole mandare un messaggio diventerà lui il client e il destinatario sarà il server. Per questo motivo i vari utenti vengono chiamati con il nome generico di *User Agent.*
Il *Proxy Server* agisce da mediatore per lo scambio dei messaggi tra UAS e UAC. Il ruolo del proxy diventa importante per consentire la gestione di messaggi in presenza di sistemi di sicurezza come i firewall che disabilitano alcune porte. Il *Redirect Server* permette agli UAC di cambiare geograficamente posizione rimanendo sempre raggiungibili con lo stesso identificativo SIP.
Il *Registrar Server,* infine, permette di alterare l'indirizzo di un utente. Questi sono i principali attori coinvolti nel protocollo e, per poter comunicare tra di loro, hanno bisogno di scambiarsi messaggi.

Fig. 2: *Attori del protocollo SIP*

MESSAGGI SIP

I messaggi che il protocollo usa sono composti da una riga iniziale dove è presente il comando, poi c'è un header con informazioni su mittente e destinatario ed infine il body del messaggio. Questa struttura è del tutto simile a quella dei messaggi HTTP.
Analizzando le righe del messaggio si possono notare chiamante e ricevente (to e from) e l'identificativo del chiamante (callid). I messaggi che gli utenti possono scambiarsi, in una sessione, sono di 6 tipi INVITE, ACK, OPTIONS, BYE, CANCEL e REGISTER e servono per gestire tutta la comunicazione. Un esempio di cosa accade si ha considerando un utente, Alice, che vuole comunicare con Bob. I due utenti usano dispositivi differenti e sono registrati presso due proxy diversi. Quello che serve è solo un nominativo con il quale l'utente è associato, come per gli indirizzi mail. In questo caso due esempi sono *sip:alice@atlanta.com* e *sip:bob@biloxi.com.* In questo caso il primo messaggio di richiesta viene effettuato da Alice con un INVITE. La request viene recapitata al proxy di Alice che le risponderà con *Trying,* ovvero un tentativo di contattare Bob. A questo punto viene eseguita una richiesta DNS, Domain Names Service, per risolvere il nome host di Bob. I proxy aggiungono il loro nome host nel campo VIA del messaggio per far sì che i messaggi SIP attraversino gli host nell'ordine indicato. In modo analogo, il messaggio passa per il proxy di Bob e, quando raggiunge l'utente, il suo telefono riceve l'INVITE e risponde con un *Ringing.*
Nell'esempio Bob risponde e viene inviato un 200 OK al softphone di Alice. In genere, all'*INVITE* di Alice viene aggiunto un body con le informazioni sul modo di comunicare con il nome della sessione, la connessione, la larghezza di banda e la descrizione del dispositivo adottato per comunicare. Appena Alice riceve da Bob OK, gli invierà ACK e la comunicazione a voce vera e propria potrà iniziare. Se Alice e Bob non dovessero avere una capacità di streaming compatibile, allora Alice dovrebbe abbandonare il protocollo con un BYE. La flessibilità del protocollo è dimostrata anche dal fatto che, una volta avviata la

TIPI DI MESSAGGIO

I principali tipi di messaggio sono 6:

INVITE, per invitare un utente in una sessione;

BYE, per terminare la partecipazione di un utente ad una sessione;

CANCEL, per cancellare una sessione;

OPTIONS, per avere informazioni sulle capacità multimediali;

ACK, per confermare;

REGISTER, per informare il Sip Server della localizzazione dell'utente.

```
              . . .atlanta.com  . . . biloxi.com. . .
            .         proxy              proxy        .
          .                                             .
 Alice's . . . . . . . . . . . . . . . . . . . . . . . . Bob's
softphone                                              SIP Phone
    |                    |                |                    |
    |     INVITE F1      |                |                    |
    |------------------->|    INVITE F2   |                    |
    |   100 Trying F3    |--------------->|     INVITE F4      |
    |<-------------------|  100 Trying F5 |------------------->|
    |                    |<-------------- |  180 Ringing F6    |
    |                    | 180 Ringing F7 |<-------------------|
    |   180 Ringing F8   |<---------------|     200 OK F9      |
    |<-------------------|    200 OK F10  |<-------------------|
    |     200 OK F11     |<---------------|                    |
    |<-------------------|                |                    |
    |                       ACK F12                            |
    |--------------------------------------------------------->|
    |                    Media Session                         |
    |<========================================================>|
    |                       BYE F13                            |
    |<---------------------------------------------------------|
    |                     200 OK F14                           |
    |--------------------------------------------------------->|
    |                                                          |
```

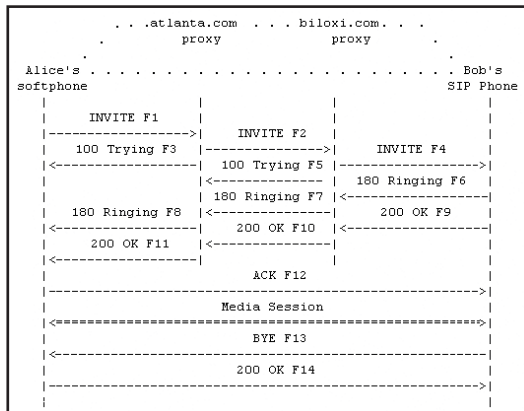



Fig. 3: ***Sequenza dei messaggi del protocollo SIP***

conversazione, Alice e Bob possano voler cambiare le caratteristiche della sessione magari utilizzando un supporto multimediale diverso e passando a quello video senza dover effettuare una nuova chiamata. Tutta questa è la teoria che c'è dietro al VoIP, ora ci si può addentrare nella pratica con le api di Java.

SIP

Il protocollo SIP, *Session Initiation Protocol*, è definito e spiegato in RFC 3261, *Request For Comment*.

JAIN

Per realizzare una chiamata su internet gli organismi per la realizzazione degli standard hanno proposto il protocollo SIP, che permette di iniziare, gestire e terminare la connessione tra due utenti. SIP, a sua volta, si appoggia ad altri protocolli, come RTP, per gestire lo scambio di dati multimediali per ottenere una buona qualità del servizio. Java si inserisce in questo contesto in una duplice veste: realizzare sia le API che implementino il protocollo SIP sia quelle per il protocollo RTP. Nel primo caso ci sono diversi pacchetti in base al dispositivo da utilizzare, nel secondo caso il riferimento è JMF, Java Media Framework.
I vari pacchetti relativi a Jain sono sviluppati da JCP, Java Community Process, e vengono classificati in progetti JSR con codice. Per sfruttare a pieno le possibilità che offre Java, le api sono orientate ai diversi dispositivi e alle piattaforme che la Sun offre. I progetti sono: 1) JSR 180 Sip per J2ME, per lo sviluppo di applicazioni mobile; 2) JSR 32 Jain-Sip per sviluppo su piattaforma J2SE; 3) JSR 116 Sip-Servlet, per piattaforma J2EE.
In realtà, a queste api complete, c'è anche una libreria aggiuntiva Jain-Sip Lite con delle classi di più alto livello che servono per astrarre dalle difficoltà e dai dettagli implementativi e consentono di sviluppare applicazioni con estrema facilità e velocità. Permette di sviluppare applicazioni di tipo User Agent, cioè terminali software che si connettono a reti SIP. Possono essere viste come involucro, Wrapper, per accedere alle librerie Jain-Sip che sono complete e richiedono uno sviluppatore che conosca il protocollo SIP. Quindi, le Jain-Sip Lite sono da consigliare per chi voglia ottenere risultati in tempi brevi. Per avere, invece, maggior controllo su cosa accade e per guidare il protocollo SIP a proprio piacimento, si deve fare riferimento alle altre api. Sip per J2ME definisce un set di interfacce per dispositivi cellulari e pda che trattano la comunicazione SIP con factory comuni alle SocketConnection e HttpConnection per semplificarne l'uso. In questo modo, infatti, si ha la gestione integrata degli header, response automatizzate e supporto ai dialoghi SIP. Sip-Servlet è un insieme di api basate sulle esistenti Servlet e funziona in maniera analoga.
Come le Servlet, devono girare in un container adatto, Servlet Container, ad esempio Tomcat. Le Sip-Servlet hanno anche l'obiettivo di rendere standard alcuni aspetti del container: l'associazione delle richieste SIP con le Servlet, il modello di sicurezza, il Servlet Deployment Descriptor (documento che accompagna la Servlet e che contiene le informazioni per il container su come deve essere gestita la Servlet) e il formato del file di distribuzione analogo al JAR Java (simile al formato WAR utilizzato dalle Servlet HTTP). In questo modo si può rispondere alle richieste SIP con interfacce semplici per realizzare sia User Agent che proxy. I dettagli di basso livello del protocollo, invece, sono demandate al Servlet Container. E' infatti il container che si occuperà di gestire le ritrasmissioni dei messaggi in caso di funzionamento come proxy, di scegliere la risposta migliore tra più risposte a seguito di richieste concorrenti, generazioni di numeri di sequenza, identificativi di chiamata e gestione degli header.
La differenza con le altre api è l'uso necessario del Servlet Container e del proxy Sip. Jain-Sip è il primo progetto proposto per la standardizzazione Java del protocollo SIP. Le interfacce sviluppate consentono la realizzazione di tutti gli attori coinvolti in una rete SIP: User Agent Client (UAC), User Agent Server (UAS), proxy, registrar server e redirect server.
Si basano su eventi e hanno vari oggetti factory per la rapida creazione di request, response e header. Dopo aver parlato del protocollo SIP per connettere gli utenti e aver visto le varie api fornite da Java, è il momento della struttura delle librerie.

INDIRIZZI

Per memorizzare gli indirizzi SIP ci sono tre alternative: URL, tipicamente usati all'interno di sistemi firewall; URI, più generici, possono essere usati anche fuori dai firewall; AOR, array di record, ancora più generici riescono a risolvere qualsiasi tipo di indirizzo.

ARCHITETTURA SOFTWARE

Un'applicazione che voglia realizzare il VoIP dovrebbe basarsi sul modello ad eventi.
In questo modo si definiscono un Listener e un Provider che scambiano, appunto, eventi. In particolare il Listener si troverà in ascolto degli eventi generati dal Provider. Gli eventi incapsulano Request e Response ed il SipListener rappresenta colui che ascolta e consuma l'evento. L'evento è un'entità astratta: quello che c'è dentro l'evento è il messaggio SIP. Il SipProvider è il fornitore dell'evento che riceve il messaggio dalla rete e lo passa all'applicazione sempre sotto forma di evento. Questo modello serve per capire come sono strutturati le interfacce delle api. Indipendentemente dal tipo di api che si usano, infatti, gli obiettivi ed i servizi offerti da Java sono: fornire metodi per formattare il messaggio in formato SIP; inviare e ricevere i messaggi; scompattare i messaggi in arrivo; gestire la transazione con timeout, stati e ciclo di vita. Resta all'applicazione il compito di implementare il SipListener per interagire con lo stack SIP; di registrarsi con il SipProvider per tutti i messaggi e gestire la transazione in modalità sincrona o asincrona; accedere agli oggetti dello stack e ricevere i messaggi dallo stack sotto forma di eventi.
Il flusso di un'applicazione dovrebbe essere come quello di **Figura 5** nella quale sono coinvolte le classi realizzate in Jain-Sip. L'interfaccia SipStack serve per gestire i Provider Sip, può registrare molti listener, è instanziato dalla SipFactory con un insieme di parametri, definisce le proprietà di ritrasmissione, le informazioni sui router ed eventuali nuovi metodi realizzati. In particolare, coordinando la ritrasmissione, solleva lo sviluppatore da una notevole complessità di gestione. I parametri da passare per il corretto funzionamento del SipStack sono il suo indirizzo IP, il nome, il path del router per l'instradamento dei messaggi prima che il dialogo sia iniziato ed il filtro per la ritrasmissione. L'interfaccia SipProvider ha invece il compito di ricevere gli eventi e notificarli al SipListener registrato e gestisce le transazioni con scambio di request e response.
L'intrerfaccia SipListener viene creata ed associata ad un unico SipStack, e tutti i SipProvider associati al SipStack hanno il medesimo SipListener; gestisce i timeout della transazione e ritrasmette gli eventi. Queste interfacce si trovano nel package generale javax.sip di Jain-Sip che definisce l'architettura generale, le transazioni e gli eventi. Gli altri package sono relativi al messaggio Sip.
Il package address contiene un wrapper per le URI, definendo interfacce per URI di tipo Sip e Tel. Gli indirizzi di un utente, infatti, possono essere come quelli di Alice e Bob, sip:alice@atlanta.com, simili a quelli di posta elettronica, oppure come numero di telefono tel:123456. Il package message definisce le interfacce per i messaggi request e response. I messaggi request contengono il tipo di richiesta e l'URI; quelli di response il codice dello stato della transazione. Il body di un messaggio può contenere anche la descrizione della sessione in formato String o Object, come specificato nel SDP, Session Description Protocol. I tipi di messaggio della request seguono il protocollo Sip e, a livello di codice, sono delle costanti.
Il package header, naturalmente, contiene tutti gli header supportati e loro possibili estensioni. Sono simili agli header http sia per sintassi che per significato. In questo package si è deciso di sviluppare ogni possibile header invece di uno solo generico che gestisse tutte le informazioni, per rendere più esplicita la gestione dei molteplici header definiti negli standard. Per poter scegliere, di

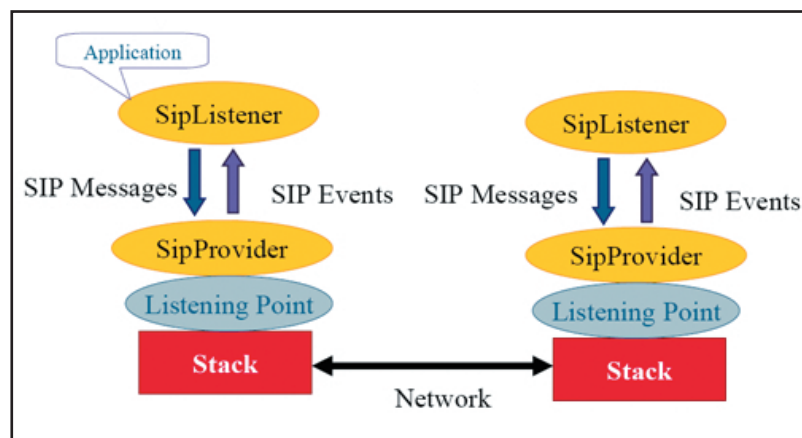



Fig. 4: *Modello ad eventi*

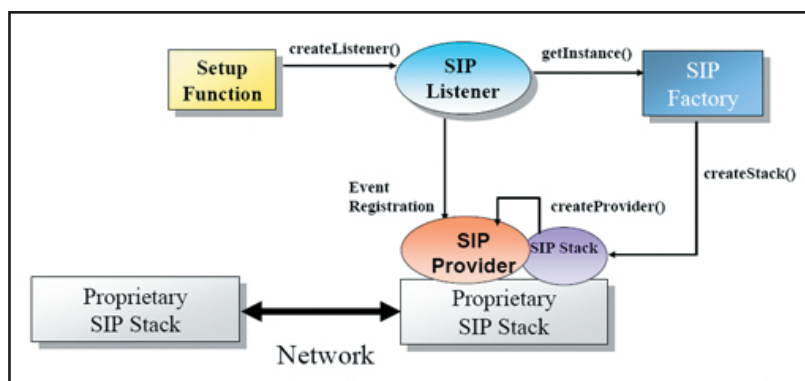



Fig. 5: *Architettura degli oggetti*

volta in volta, tra le diverse alternative, ci sono quattro classi che implementano il pattern factory: AddressFactory, HeaderFactory, MessageFactory e SipFactory per creare, come suggerisce il nome, indirizzi URI sip e tel, gli header, i messaggi con request e response e gli oggetti Stack. I vantaggi di usare questa suddivisione sono la facilità di aggiungere nuovi metodi e header.

Una volta capito classi e interfacce che regolano la comunicazione si può pensare di creare un'applicazione simile a quelle che si trovano, scaricabili anche gratuitamente, su internet.

RTP

Il protocollo RTP, -Time Transport Protocol, è definito e spiegato in RFC 1889, Request For Comment.

APPLICAZIONE

Un'applicazione che voglia far comunicare due utenti si definisce, con termine inglese, Third Party Call Control, 3PCC, e cioè un applicazione che permetta a due utenti qualsiasi di colloquiare.

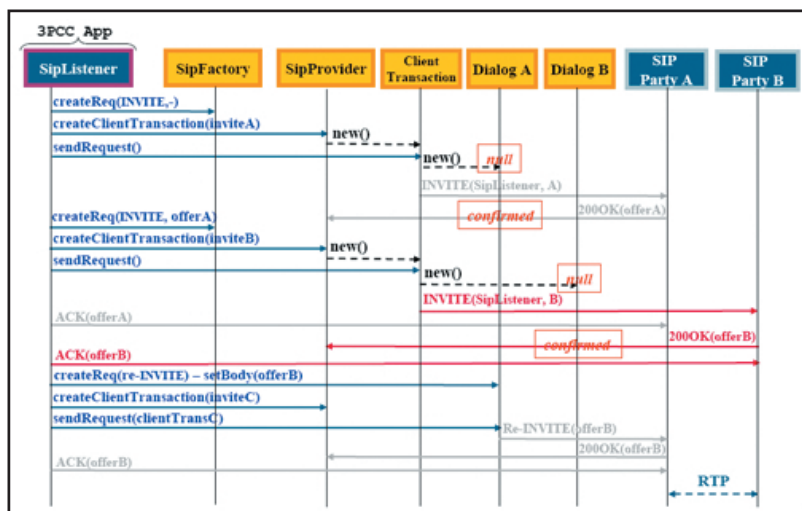



Fig. 6: *Sequenza di un'applicazione*

In **Figura 6** si apprezza tutto il passaggio delle richieste. Per completezza, il dialogo è un'associazione peer to peer, punto punto, tra due utenti che stabiliscono una connessione Sip e rappresenta, in maniera astratta, il contesto nel quale interpretare i messaggi Sip. Si può osservare che tra i due utenti si stabilisce una connessione RTP, della quale tratteremo a breve. Prima alcuni dettagli di codice per capire quanto sia facile realizzare un'applicazione del genere.

Vi è l'inizializzazione dello stack, usando la SipFactory e impostando i parametri dello stack.

```
try {
  Properties properties = new Properties();
  properties.setProperty("javax.sip.IP_ADDRESS",
                          "129.8.9.78");
  //inizializzazione altre proprietà
} try
{
  sipStack = sipFactory.createSipStack(properties);
} catch(SipException e)
{
  System.exit(-1);
}
}
```

L'inizializzazione della request specificando, ad esempio, la URI e il tipo di messaggio (invite).

```
try {
  SipURI requestURI = addressFactory.createSipURI
                        (toUser, toSipAddress);
  //Creazione altri header
  Request request = messageFactory.createRequest
          (requestURI, Request.INVITE, callIdHeader,
              cSeqHeader, fromHeader, toHeader,
                    viaHeaders, maxForwards);
}
```

L'invio del messaggio di request usando una ClientTransaction.

```
try {
  //Creazione ClientTransaction
  ClientTransaction inviteTid =
      sipProvider.getNewClientTransaction(request);
//Invio request
sipProvider.sendRequest(inviteTid,request);
}
```

E la gestione dei messaggi come eventi usando una ServerTransaction.

```
try
{
  public void  processRequest(
                       RequestEvent requestEvent)
  {
    Request request =
            requestReceivedEvent.getRequest();
    ServerTransaction st =
            requestEvent.getTransaction();
    //gestione specifica della request
  }
}
```

Abbiamo visto le Jain-Sip api ed un loro possibile utilizzo in un'applicazione. Ora resta "solo" la gestione della qualità: la voce non può essere trattata come i dati e allora serve

un protocollo che garantisca una certa priorità ai pacchetti vocali.

RTP E JMF

RTP, Real-Time Transport Protocol, è il protocollo per il trasferimento di pacchetti con contenuto mutlimediale come audio e video. Poiché l'obiettivo è di garantire la Qualità del Servizio, RTP si usa in maniera congiunta con RTCP, Real-Time Transport Control Protocol, che si occupa di monitorare la qualità dei dati trasmessi aumentando o anche diminuendo la banda dedicata alla trasmissione. E' pensato per lavorare in modo indipendente dal livello di trasporto anche se, in genere, usa UDP, User Datagram Protocol e IP, Internet Protocol. Non si può utilizzare TCP per l'eccessiva pesantezza dei pacchetti e con UDP non è possibile controllare l'ordine di arrivo degli stessi. Ecco, allora, che RTP risolve il problema permettendo al ricevitore di organizzare i pacchetti in base all'ordine di arrivo, trascurando i pacchetti persi. Per realizzare il protocollo RTP, Java si appoggia a JMF, Java Media Framework, che fornisce un'architettura unificata per la cattura, emissione e manipolazione di dati multimediali. JMF permette la riproduzione e trasmissione di stream RTP attraverso l'uso di classi e interfacce definite in javax.media.rtp, javax.media .rtp.event, e *javax.media.rtp.rtcp* che sfruttano il modello ad eventi. Mettendo insieme tutti i tasselli del puzzle, SIP si occupa di trovare e stabilire una connessione tra gli utenti, specificando le caratteristiche della comunicazione grazie al SDP, Session Description Protocol presente nel body del messaggio. RTP permette di inviare lo stream dei pacchetti della voce e RTPC ne controlla la qualità. JMF ha il compito di definire la sorgente dalla quale prelevare la voce, convertirla da formato analogico a digitale, per poterla avere come flusso di byte e stabilire il protocollo RTP verso il destinatario. JMF necessita di tutti i dati relativi all'indirizzo dell'utente al quale inviare lo stream della voce che sono contenuti nel body del messaggio SIP, conformi al SDP. Sviluppare un'applicazione vuol dire creare un oggetto che permetta, prima di stabilire una connessione SIP tra utenti e prendere i dati relativi ad un tipo di streaming usando Jain-Sip, e poi chiamare JMF per iniziare il protocollo RTP creando così la sequenza dei pacchetti voce su internet. Si deve quindi costruire un oggetto con le api JMF cha cattura la voce, la codifica in formato digitale opportuno, campionandola ad esempio alla frquenza di 44kHz e la associa ad una lista di dispositivi audio.

```
Vector audioDevices =
        CaptureDeviceManager.getDeviceList(new
   AudioFormat(AudioFormat.LINEAR, 44100, 16, 2));
```

A questo punto si apre il canale di stream per la voce usando i parametri SDP come indirizzo e porta del destinatario.

```
public void openVoiceStream (String sdpData)
                              throws MediaException
{
  …
  SessionDescription  sessionDescription =
      sdpFactory.createSessionDescription(sdpData);
  Connection sessionConnection =
                    sessionDescription.getConnection();
  //avvio oggetti per inviare e ricevere lo stream voce
}
```

Gli oggetti per inviare e ricevere lo stream voce possono essere realizzati prendendo spunto dagli esempi forniti con JMF. Il loro compito è di scambiare il flusso della voce utilizzando le classi JMF PushBufferStream e PullBufferStream per sfruttare le capacità offerte dal Buffer, PushBufferDataSource per catturare il flusso dati e RTPManager e SendStream per l'invio. Quindi le api JMF si usano per creare una applicazione client /server che apre una connessione e invia uno stream. Riassumendo, SIP eRTP, realizzati dalle api Jain-Sip e JMF si uniscono e completano a vicenda per localizzare gli utenti, stabilire le caratteristiche della trasmissione ed effettuarla materialmente.

CONCLUSIONI

Per capire gli obiettivi, i problemi e la realizzazione del VoIP, la telefonia per mezzo di internet, abbiamo parlato dell'architettura, degli standard che sono stati proposti e realizzati e come utilizzarli per sviluppare le applicazioni che stanno prendendo piede nel mondo dell'informatica. In conclusione, considerando anche che i protocolli ed i modelli adottati sono assai simili a quelli utilizzati in qualsiasi applicazione internet, la struttura e le api viste possono essere prese come esempio per sviluppare sia queste nuove telefonate, sia le applicazioni che comunemente utilizziamo.

Cristiano Bellucci

Web Services con semplicità

In questo articolo illustreremo il protocollo XML-RPC. Utilissimo per applicazioni che fanno uso della programmazione distribuita, ma meno complicato di Soap. Vedremo, infine, un esempio d'uso in PHP

Recentemente l'obiettivo principe di tutti i linguaggi di programmazione è la "cooperazione". La presenza di Internet fa si che enormi quantità di codice atto a risolvere i problemi più disparati sia presente in rete in maniera isolata. Ad esempio il programmatore *"A"* avrà sviluppato una funzione che calcola il codice fiscale di una persona sulla base dei suoi dati anagrafici, contemporaneamente qualche altro migliaio di programmatori avranno già sviluppato la stessa funzione, e ciascuna funzione vive in un contesto isolato, serve un unico programma. Lo spreco di risorse è immenso, l'ideale sarebbe non dovere sviluppare tutte le volte una nuova funzione ma potere riutilizzare il codice sviluppato da altre persone e creare su Internet una sorta di grande framework che espone funzioni utili a tutti. La realizzazione di questo obiettivo è al di la dall'essere lontana. In questo articolo analizzeremo una fra le tecniche disponibili.

DIVERSI TIPI DI STRUTTURA

L'idea di base per la creazione di architetture distribuite è la *"Remote Call"*, ovvero la possibilità di concedere ad un software di richiamare una funzione esposta da un'applicazione collocata in una posizione geografica molto distante da quella della macchina in cui è in esecuzione.
Per mettere in collegamento i due applicativi si è scelto si utilizzare il protocollo TCP/IP e quindi ovviamente Internet. Ciò che più è straordinario però è che al di la del protocollo di trasporto, ad un livello più alto si è scelto di fare dialogare i due estremi per mezzo di HTTP, ovvero il protocollo applicativo che viene utilizzato per la trasmissione delle pagine WEB. Niente di più semplice dunque. A questo punto rimaneva ancora da stabilire un linguaggio comune sul quale le richieste e le risposte potessero intendersi. Ovvero, se la macchina *A* richiama una procedura presente sulla macchina *B*, è evidente che *B* deve essere in grado di interpretare la richiesta e a sua volta deve potere rispondere con un formato comprensibile da *A*.
Si è scelto di effettuare questo passaggio di informazioni tramite XML. E qui sono cominciati i guai. Di fatto XML consente di definire dei propri standard, per cui se era chiaro che il formato delle risposte/richieste doveva essere XML non era ancora chiaro come il documento XML dovesse essere composto. Per questo motivo il W3C ha formato una commissione controllata dalle maggiori case produttrici di software fra cui Microsoft, Lotus, IBM e molte altre, da questa commissione è nato SOAP, ovvero il formato XML su cui si basa la possibilità di richiamare procedure remote.
Tuttavia, e qui la storia si fa complicata perché non è ben chiaro chi ha inventato cosa, SOAP è nato da un protocollo preesistente l'XML-RPC e vi ha aggiunto una serie di funzionalità che a detta di molti hanno sporcato il protocollo originario, rendendolo più complesso e meno efficace del necessario.
In questo articolo noi ci occuperemo di XML-RPC e non di SOAP. Prima di tutto perché i due protocolli sono realmente molto simili, in secondo luogo perché XML-RPC non ha la complessità di SOAP pur assolvendo alle stesse funzioni, in terzo luogo proprio perché XML-RPC recentemente è venuto alla ribalta poiché usato da una grande varietà di Blog per implementare il così detto meccanismo di *TrackBack*.

XML-RPC IN PRATICA

Abbiamo già detto che le chiamate ad una procedura remota e le conseguenti risposte verranno effettuate utilizzando XML e il protocollo XML-RPC. Allo stesso modo abbiamo detto che la chia-

mata sarà inoltrata ad un'applicazione web che gira sulla porta 80 di una macchina e tramite il tradizionale protocollo HTTP. Al momento non ci preoccuperemo di come la chiamata viene inoltrata, ci preoccuperemo di come debbano essere formalizzate le richieste e le risposte in linguaggio XML. Ad esempio il seguente spezzone di codice rappresenta una chiamata alla procedura remota *getStateName()* alla quale viene passato come parametro il valore 41

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181
<?xml version="1.0"?>
<methodCall>
  <methodName>getStateName</methodName>
  <params>
      <param>
       <value><i4>41</i4></value>
       </param>
   </params>
  </methodCall>
```

la chiamata si compone di due parti: l'header e il payload. Nell'header troviamo. L'User-Agent e l'host che stanno effettuando la chiamata, il content type *text/xml* e il *content-lenght* che segna la lunghezza del messaggio. Il *payload* è formato da una struttura il cui nodo radice è *<methodCall>* che indica che stiamo effettuando una richiesta. All'interno di questo nodo troviamo *<methodName>* che contiene il metodo da richiamare. Ancora più internamente nella gerarchia troviamo *<params>* al cui interno possono essere contenuti uno o più valori *<param>* ciascuno valorizzato con un *<value>*. Sicuramente tutta questa spiegazione sembrerà prolissa, ma se date un'occhiata alla struttura XML vi risulterà semplice e piuttosto chiara. Qualche annotazione deve essere fatta per quanto riguarda la valorizzazione dei parametri. Di fatto è possibile indicare un "tipo" che specifica ulteriormente il senso del valore del parametro. **(Tabella 1)** Un particolare significato assume invece il tipo *<struct>*

```
<struct>
  <member>
    <name>lowerBound</name>
    <value><i4>18</i4></value>
    </member>
  <member>
    <name>upperBound</name>
    <value><i4>139</i4></value>
    </member>
  </struct>
```

Che come potete vedere assume il significato di una sorta di record al cui interno sono contenute delle celle che hanno un nome e un valore che a sua volta può anche essere di tipo *struct*. Infine l'ultimo tipo strutturato che può essere passato come parametro ad una procedura di remote *call* è il tipo array:

```
<array>
  <data>
    <value><i4>12</i4></value>
    <value><string>Egypt</string></value>
    <value><boolean>0</boolean></value>
    <value><i4>-31</i4></value>
    </data>
  </array>
```

Il cui significato dovrebbe essere abbastanza chiaro, senza ulteriori commenti. Manca ovviamente il nome delle celle. In realtà, come vedremo quando implementeremo una remote call in PHP, la maggior parte di queste informazioni non ci serviranno. È corretto sapere che nella sua formalizzazione più rigorosa, XML-RPC si comporta come illustrato. Di fatto volendo utilizzare il protocollo con qualche altro linguaggio, il tutto sarà abbastanza utile.

<i4> o <int>	Un intero di 4 byte	*-12*
<boolean>	0 (false) or 1 (true)	*1*
<string>	string	hello world
<double>	numeri in virgola mobile	*-12.214*
<dateTime.iso8601>	date/time	*19980717T14:08:55*
<base64>	base64-encoded	*binaryeW91IGNhbid0IHJlYW QgdGhpcyE=*

Tabella 1: Alcuni dei tipi disponibili per la struttura XML

LA RISPOSTA

Se è chiaro come avviene una chiamata ad una procedura remota, deve essere altrettanto chiaro come il server remoto restituisce l'esito al chiamante, utilizzando ancora una volta il formato XML. La struttura di una risposta si compone come sempre di un header e di un payload.

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 426
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:02 GMT
Server: UserLand Frontier/5.1.2-WinNT
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
```

```
        <struct>
          <member>
            <name>faultCode</name>
            <value><int>4</int></value>
            </member>
          <member>
            <name>faultString</name>
            <value><string>Too many parameters.
                                    </string></value>
            </member>
          </struct>
        </value>
      </fault>
    </methodResponse>
```

Per quanto riguarda l'analisi del XML invece tipicamente la root *<methodResponse>* contiene semplicemente un *<params>* che a sua volta contiene un *<param>* che a sua volta contiene ancora un singolo *<value>* che definisce il valore della risposta. Viceversa il *<methodResponse>* potrebbe anche contenere un *<fault>* come nel nostro esempio che a sua volta contiene dei dati che identificano un eventuale problema. In unica risposta XML non possono convivere segmenti *<fault>* e segmenti *<params>*. Se una richiesta non ha dato esito corretto, ovviamente non può esserci un *params* che identifica il risultato.

VULNERABILITÀ DI XML-RPC

Nell'implementare le proprie applicazioni bisogna stare attenti ai vari aspetti legati alla sicurezza. Recentemente un bug dell'implementazione XML-RPC di Drupal, una delle piattaforme di blogging più note al mondo, ha comportato una serie di rischi per gli utenti, specialmente in relazione alla fun-zione di trackback. La colpa non era ovviamente del protocollo, ma di una falla nell'implementazione.

L'IMPLEMENTAZIONE PHP

Per i nostri scopi utilizzeremo un pacchetto Pear che ci consente di utilizzare degli oggetti che implementano già tutta la trasmissione/ricezione di XML-RPC senza dover reinventare l'acqua calda. Inizieremo implementando una procedura che restituisce la stringa *"Hello World"* se gli viene passato il parametro *"1"* oppure restituisce la stringa *"Bye Bye"* se gli viene passato il parametro *"2"*. Il file php che farà da server si troverà sull'Host 192.168.1.2 e si chiamerà *xml-rpc.php*. Ovviamente diamo per scontato che su quell'host ci sia un server web correttamente configurato. Il nostro *xml-rpc.php* conterrà il seguente codice:

```
<?php
require_once 'XML/RPC/Server.php';
function sayme($params) {
    $param = $params->getParam(0);

    if (!XML_RPC_Value::isValue($param)) {
        return $param;}
    switch ($param->string) {
        case $param=="1":
              $val = "Hello World";
              break;
        case $param=="2":
              $val = "Bye Bye";
              break;
         default:
              $val = "Ok";
    }
    $response = new XML_RPC_Value($val);
    return new XML_RPC_Response($response);
  }
  $server = new XML_RPC_Server(
     array( 'tellme' =>array('function' => 'sayme'
       )
     )
);
?>
```

Nella prima linea viene incluso il file Pear che contiene la logica per la gestione XML-RPC. La procedura che effettua fisicamente le operazioni è la *sayme()* che prende in input un parametro. Al suo interno prima di tutto recupereremo i valori passati alla funzione tramite il metodo *getParam()*. Con un semplice *if* controlliamo se il valore passato è accettabile o in qualche modo la sua definizione non corrisponde a quella aspettata. Con uno *switch* verifichiamo il valore passato. Notate che per questa verifica abbiamo usato *$param->string*, in effetti il valore restituitoci da *params->getParam(0)* è ancora a sua volta un oggetto, per cui dobbiamo reperire il suo valore con un metodo opportuno. Se avessimo voluto prelevare dal parametro un valore "Intero" piuttosto che una stringa avremmo dovuto usare *$param->scalarval()*. Una volta ottenuti i risultati voluti, non possiamo restituirli direttamente, ma dobbiamo convertirli in un oggetto di classe *XML_RPC_Response*. Il costruttore di questo oggetto accetta come parametro un oggetto di tipo *XML_ RPC_Value*. Per cui le due righe

```
  $response = new XML_RPC_Value($val);
    return new XML_RPC_Response($response);
```

assolvono la prima alla funzione di convertire la *$val* in un oggetto di tipo *XML_RPC_Value*, la seconda di restituire al chiamante un oggetto di classe *XML_RPC_Response*.
Infine le righe

```
$server = new XML_RPC_Server(
    array('tellme' =>array(
       'function' => 'sayme'
    )
  )
);
?>
```

Inizializzano il server *xml-rpc* e associano la fun-

zione 'virtuale' *tellme()* alla funzione 'fisica' *sayme()*. Significa che quando il client invocherà la funzione remota *tellme()*, il server si preoccuperà di mappare questa funzione su quella in essa contenuta *sayme()*.

IL CLIENT

Il nostro Client si chiamerà, con un grande sforzo di fantasia *client.php* e conterrà il seguente codice:

```
<?
require_once 'XML/RPC.php';
$params = array(new XML_RPC_Value(1, 'string'));
$msg = new XML_RPC_Message('tellme', $params);
$cli = new XML_RPC_Client('/xml-rpc.php', 'localhost');
$cli->setDebug(1);
$resp = $cli->send($msg);
if (!$resp)
{
    echo 'Communication error: ' . $cli->errstr;
    exit;
}
if (!$resp->faultCode())
{
    $val = $resp->value();
    $data = XML_RPC_decode($val);
    echo $data;
} else
{
    echo 'Codice errore: ' . $resp->faultCode() . "\n";
    echo 'Descrizione: ' . $resp->faultString() . "\n";
}
?>
```

Al solito, la prima riga è chiara. Include il codice pear necessario. Con la riga successiva viene creato un array di oggetti *XML_RPC_Value* .Nel nostro caso si tratta di un array con un solo elemento, di fatto dobbiamo passare un solo parametro. Il costruttore della classe *XML_RPC_Value* accetta come parametro, il valore da passare e il suo tipo.
Successivamente inizializziamo un oggetto di classe *XML_RPC_Message*. Sarà questo oggetto a formattare il messaggio XML che deve essere inviato al server. Il costruttore accetta come valori, la funzione remota da richiamare, nel nostro caso la *tellme*, e un array di oggetti di tipo *XML_RPC_VALUE*. Infine creiamo l'oggetto *xml_rpc_client*, passandogli come valori l'host che contiene il server e il nome del file in cui il server viene definito. Settiamo il Debug del client a 1 per analizzare i risultati ottenuti. In fase di produzione questo parametro deve essere settato a 0, altrimenti avrete sempre in output il messaggio XML di risposta. E per finire mandiamo tutto al server con il metodo *send* dell'oggetto *XML_RPC_Client*. Il valore restituito sarà contenuto nella variabile *resp* che a sua volta è un oggetto di tipo *XML_ RPC_Response*.

L'ANALISI DELLA RISPOSTA

Qui il codice è abbastanza semplice, si affida tutto a:

```
$val = $resp->value();
$data = XML_RPC_decode($val);
echo $data;
```

Il valore restituito viene prelevato tramite il metodo value e decodificato tramite il metodo *XML_RPC_decode*. I vari if non sono altro che metodi di controllo che ci serviranno nel caso in cui vengano restituiti codici di errore. L'architettura client /Server basata su XML-RPC è servita!

CONCLUSIONI

XML-RPC è una buona alternativa quando non si vuole entrare nei meandri di SOAP. Si tratta di un protocollo abbastanza semplice e affidabile. La sua validità è testimoniata anche dalla centinaia di applicazioni che ne fanno uso, soprattutto come meccanismo di base per l'interconnessione dei blog. Si tratta di una tecnica che in congiunzione a PHP e PEAR non richiede particolari sforzi per essere implementata pur rimanendo intatta in tutta la sua potenza. Utile in più di una situazione dunque.

SUL WEB

Http://www.xmlrpc.com
Contiene tutte le specifiche del protocollo oltre ad esempi e tutorial

http://pear.php.net
Contiene la documentazione sul package XML-RPC

COSA È PEAR E COME INSTALLARLO

Con la sigla PEAR si definisce un insieme di classi, il cui sviluppo è fortemente sostenuto dal PHP user Group e che estendono notevolmente le funzioni del linguaggio. Il repository di riferimento per le classi Pear è *pear.php .net.* L'idea interessante è che le classi Pear sono sviluppate in PHP, per cui si tratta di normali file php che possono essere inclusi nei vostri progetti con le direttiva require, *require_once* o include a seconda delle necessità. Nonostante questo, data la complessità della loro costruzione, in alcuni casi potrebbero dipendere le une dalle altre, il che comporterebbe la ricerca e il download manuale dei vari file. Per agevolare l'uso delle classi pear esistono dunque dei comandi specifici che evitano un intervento manuale. In particolare, in ambiente linux, per installare il gestore dei download il comando è

```
lynx -source http://go-pear.org/ | phpin
```

ambiente windows

```
php go-pear.bat
```

il file go-pear.bat è contenuto nella directory di installazione del PHP. In tutti i casi per l'installazione dei package pear necessari, il comando è:

```
pear install <package>
```

dove *package* è il nome del pacchetto

ASP.NET 2.0 Le master Pages

Alla scoperta di una delle novità più interessanti contenute nella nuova versione del Framework di Microsoft. Una tecnica che ci consente di risparmiare tempo e guadagnare in flessibilità

Il problema è noto: in ogni sito Web ci sono parti statiche e parti dinamiche. Ad esempio l'header di una pagina è tipicamente una parte statica, il contenuto informativo sotto l'header è tipicamente una parte dinamica che varia in continuazione.
Le *Master Pages* vengono introdotte in ASP.NET 2.0 per sopperire alla carenza di metodi semplici e pratici per la definizione di un layout di pagina efficace. In passato si sono adottate diverse tecniche per realizzare graficamente un sito web che rispettasse un certo formato grafico. Ad esempio, il classico layout di un portale, con intestazione e pié di pagina, colonne laterali e contenuto centrale poteva essere realizzato in vari modi. Ricopiando in ogni pagina le stesse istruzioni HTML oppure con delle istruzioni di *include* di altre sotto-pagine ASP che al momento della visualizzazione venivano inserite all'interno della pagina principale. Entrambi i metodi sono poco pratici e di difficile gestione. Il primo, crea una notevole mole di lavoro se si presenta la necessità di effettuare una modifica. Ad esempio se cambia l'intestazione del sito, ogni pagina deve essere ritoccata. Il secondo metodo ha i suoi svantaggi, soprattutto dal punto di vista della costruzione grafica della sotto-pagina ASP.

MASTER PAGES

La soluzione in ASP.NET 2.0 si chiama *Master Pages*. È una pagina ASPX a tutti gli effetti, può contenere controlli, immagini, tabelle, ecc. proprio come una pagina normale. Ha però due sostanziali differenze:

1. Il nome della pagina deve terminare con l'estensione *.master*
2. Il codice della pagina deve contenere come prima riga l'istruzione *<% @master %>*.

La prima differenza permette al sistema di evitare che la pagina possa essere richiamata direttamente dal browser. La seconda caratteristica permette di specificare al motore ASP.NET che si tratta di una pagina master. Tramite alcuni attributi, inoltre, è possibile specificare il linguaggio da utilizzare oppure un'altra pagina master da inserire in una pagina padre, utile per creare sottopagine statiche.
Vediamo un esempio:

```
<%@ Master Language="C#" %>
```

Una volta creato lo scheletro che costituisce il layout del sito è possibile definire una o più zone all'interno della master page destinate a visualizzare un contenuto dinamico, che varia di pagina in pagina. Questo spazio è definito tramite un nuovo controllo web chiamato *ContentPlaceHolder* che deve essere necessariamente inserito all'interno della master pages. Per definire un nuovo *ContentPlaceHolder* all'interno della pagina master occorre scrivere la seguente istruzione:

```
<asp:ContentPlaceHolder id="cphLeft"
                                    runat="server" />
```

L'identificativo id, che deve essere univoco, serve per le pagine di contenuto che utilizzeranno il layout definito dalla pagina master, mentre il classico attributo runat definisce che il controllo sia accessibile lato server, e quindi richiamabile da codice tramite l'identificativo univoco.

VISUAL WEB DESIGNER EXPRESS

Vediamo come attraverso il tool di sviluppo gratuito di Microsoft, *Visual Web Developer Express*, scaricabile dal sito *http://www.asp.net* sia possibile creare facilmente una pagina master. Dopo aver mandato in esecuzione *Visual Web Designer Express Edition* occorre scegliere la voce *New Web Site...* dal menu

File. Dalla dialog box che comparirà sarà necessario scegliere il tipo di progetto da creare. Scegliamo *Empty Web Site* come template, indichiamo una cartella dove inserire il progetto e diamogli un nome a piacere. Dalla combo box *Language* scegliamo il linguaggio preferito con cui programmare il sito.

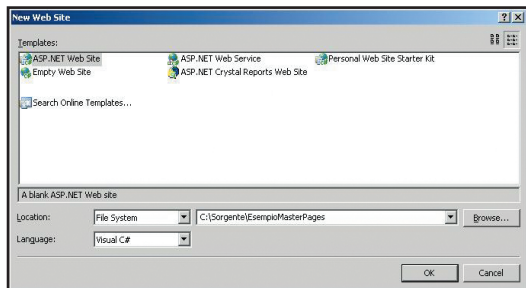



Fig. 1: ***La dialog box che ci consente di scegliere il progetto da creare***

Dal menu *Website* scegliamo la voce *Add Item...* nella nuova dialog scegliamo il template *Master Page* e cambiamogli il nome in *Default Master*. A questo punto disponiamo di una *Master Page*. È necessario associare una *"Web Form"* alla Master Page per potere inserire il codice. Una *Web Form* è sostanzialmente un contenitore per la parte dinamica del sito. Fisicamente si tratta di una normale pagina Aspx che conterrà al suo interno qualcosa del genere:

```
<%@ Page Language="VB"
                    MasterPageFile="~/default.master"
   AutoEventWireup="false"
   CodeFile="Default.aspx.vb"
   Inherits="_Default" Title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID=
                    "id="cphLeft" runat="Server">
   <h1>mainContent</h1>
        Hello World
</asp:Content>
```

Notate che è presente sia il riferimento alla Master Page sia al *placeholder*. Per aggiungere la Web Form clicchiamo su *"Add Item"* e selezioniamo il template Web Form. Noterete che si attiverà il check box *Select master page* che ci consentirà di specificare quale è la Master Page dalla quale ereditare il layout.

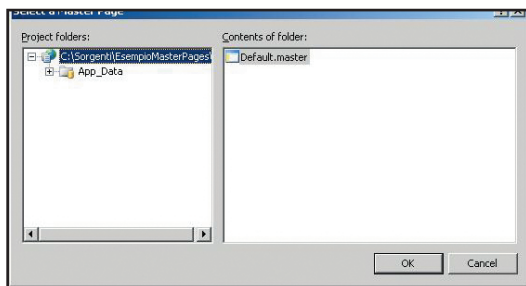



Fig. 2: ***La dialog box ci permette di associare la pagina ASP.NET alla master page***

A questo punto con un po' di semplici operazioni possiamo aggiungere un'intestazione e un pié di pagina all'interno della pagina Master.

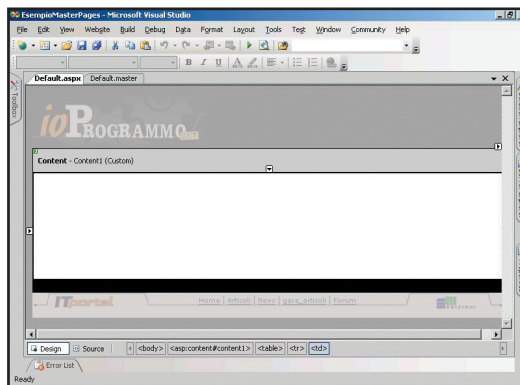



Fig. 3: ***I nostro progetto in fase di design si presenta come in figura***

DAL DESIGN ALL'ESECUZIONE

Visual Web Designer visualizza in maniera trasparente tutti i controlli, le immagini e le scritte che sono state inserite all'interno della pagina Master permettendo di concentrarsi sul contenuto della pagina stessa. Premendo *CTRL+ F5* o selezionando il menu *Start Without Debugging* dal menu *Debug* mandiamo in esecuzione il programma. Un'istanza di Internet Explorer si attiverà mostrandoci la pagina dichiarata come pagina iniziale all'interno dell'ambiente di sviluppo (all'interno della finestra *Solution Explorer*, premendo con il tasto destro del mouse sul nome di una pagina ASPX è possibile dichiararla pagina iniziale selezionando *Set As Start Page*). In **Figura 4** è visibile il sito in esecuzione. La pagina finale viene ricomposta dal motore ASP.NET in modo da unire le informazioni della pagina Master con quelle della pagina *Content*. Sempre in

LA PROPRIETÀ MASTER

La classe Page definita all'interno del .NET Framework possiede una proprietà *Master* che permette di accedere, dal codice della pagina *Content*, agli oggetti e alle proprietà della pagina *Master*. Ad esempio se si volesse cambiare il titolo della pagina definito genericamente dalla pagina *Master*, si potrebbe scrivere questo semplice codice all'interno della pagina *Content*:

```
protected void Page_Load(
            object sender, EventArgs e)
{ Master.Page.Header.Title =
                        "Nuovo Titolo";}
```

Tramite la proprietà *Page* è possibile accedere alla proprietà *Header* che, a sua volta, fornisce una proprietà *Title* che permette di cambiare il titolo della pagina. Ci sono altre proprietà interessanti fornite da *Master*. Ad esempio per cambiare i metadata di una pagina:

```
Master.Page.Header.Metadata.Add(
   "Description","Calcio,Sport,Motori");
```

In questo caso la collezione *Metadata* permette di aggiungere metadata come la *Description* utile per i spider dei motori di ricerca.

UN SITO PERSONALE IN 5 MINUTI

Oltre ai classici progetti per creare un sito o un servizio ASP.NET esiste una curiosa novità: il *Personal Web Site Starter Kit*. Questo progetto crea a tutti gli effetti un nuovo sito personale dove poter inserire le proprie foto, parlare di sé, ecc. Un ottimo modo per cominciare a districarsi tra le novità di ASP.NET 2.0.

Figura 4 è evidenziata un'altra interessante novità offerta dal nuovo ambiente di sviluppo: *ASP.NET Development Server.* Senza dover necessariamente installare sul proprio computer *Internet Information Server* e tutto quanto il resto, necessario per mandare in esecuzione un sito ASP.NET, questo modulo fornito dall'ambiente di sviluppo si occupa di interpretare e mandare in esecuzione il sito garantendo gli stessi risultati di un server Web.

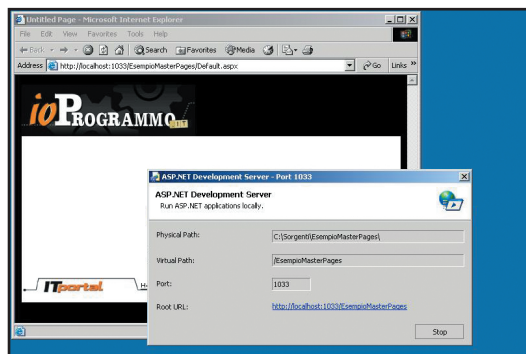



Fig. 4: ***La pagina viene ricomposta per essere visibile in Internet Explorer senza avere bisogno di IIS***

PICCOLI BUG

Una nota critica da fare all'ambiente di sviluppo è che non permette di creare e gestire graficamente le *master pages* innestate. Provando a passare dalla modalità sorgente a quella di design si riceve una notifica che avverte che la funzione non è suportata.

MASTER PAGES INNESTATE

Abbiamo accennato, precedentemente, la possibilità di innestare altre master pages all'interno di una master page principale. Questo può essere utile nel caso in cui il sito debba presentare un layout che varia in base alla sezione selezionata dall'utente. Immaginiamo di dover realizzare un sito sportivo, con varie sezioni per ogni disciplina. Il nome del sito, l'intestazione e il pié di pagina rimarrebbero uguali nel corso di tutto il sito ma le varie sezioni potrebbero visualizzare immagini, stili e colori differenti basandosi sulla sezione corrente. Per realizzare una pagina master innestata occorre aggiungere l'attributo *MasterPageFiles* e specificare il nome del file *.master* a cui agganciarsi. Vediamo un esempio:

```
<%@ Master %>
<html>
<head>
<title>Sportissimo</title>
</head>
<body bgcolor="Black">
<form id="form1" runat="server">
<h1>Benvenuti nel sito sportivo...</h1>
<asp:contentplaceholder id="cphMaster"
                                        runat="server" />
</form>
</body>
</html>
```

Questo è il codice di una semplice master page. Al suo interno viene definito uno spazio per contenere una pagina *Content* che in realtà verrà usato da un'altra Master page.

```
<%@ Master MasterPageFile="~/Principale.master" %>
<asp:content ContentPlaceHolderID="cphMaster"
                                        runat="server">
<table width="100%" bgcolor="White">
<tr>
<td colspan="2">
<h1>Calcio</h1>
</td>
</tr>
<tr>
<td>
<img src="images/calcio.gif" />
</td>
</tr>
</table>
<table width="100%" bgcolor="Black">
<tr>
   <td>
   <asp:contentplaceholder id="cphContenuto"
                                        runat="server" />
            </td>
</tr>
</table>
</asp:content>
```

La pagina Master innestata fa riferimento a quella principale tramite l'attributo *MasterPageFile.* Poi, grazie al controllo *<asp:Content>* e al suo identificativo è possibile associare la pagina master allo spazio riservato dal *ContentPlaceHolder* della master page principale.

```
<%@ Page MasterPageFile=
                        "~/MasterInnestata.master" %>
<asp:Content ContentPlaceHolderId="cphContenuto"
                                        Runat="Server">
Notizie sportive di Calcio
</asp:Content>
```

CONCLUSIONI

Le Master Pages sono senz'altro una delle più interessanti novità introdotte da ASP.NET 2.0. Grazie al completo supporto grafico e di disegn da parte di Visual Web Designer 2005 Express è possibile creare le pagine di un sito, lavorando con le Master Pages, vedendo in maniera chiara e funzionale, lo sviluppo grafico della pagina. Infine, con gli oggetti messi a disposizione dal nuovo .NET Framework è possibile realizzare anche funzionalità non immediatamente realizzabili a design-time, come il caricamento di pagine Master a run-time o il cambiamento di caratteristiche grafiche, di testo, ecc.

Fabio Claudio Ferracchiati

Super Database con ADO.NET

Il framework di Microsoft è quasi perfetto, ma noi saremo capaci di migliorarlo ulteriormente creando classi in grado di salvare direttamente su DB immagini, file e persino interi oggetti

Chi ha avuto modo di studiare attivamente .NET avrà certamente apprezzato la dovizia con la quale sono state progettate ed implementate le parti più critiche del framework. Ma qualcosa è stato trascurato. Un esempio per tutti è certamente il potente supporto allo streaming implementato nel namespace *System.IO*. L'architettura di streaming offre una classe base astratta *Stream* e due implementazioni reali basate sulla memoria *(MemoryStrem)* e sui file *(FileStream)*, ma manca del tutto qualcosa che si basi su ADO.NET e quindi, che usi i database come sistema di persistenza delle informazioni binarie. E questo è davvero un peccato perché il framework consentirebbe la persistenza anche di oggetti e quindi di intere istanze di programma recuperabili, quando necessario, direttamente a partire dal sistema di memorizzazione dei dati per eccellenza: i database, se solo questo fosse possibile direttamente…

Purtroppo non lo è, ed ecco dunque la nostra missione della giornata: progettare e scrivere il pezzo mancante.

STREAMING SU ADO.NET

La classe *Stream* espone metodi che leggono e scrivono vettori di byte; infatti qualsiasi stream, indipendentemente da quale sia la sua struttura e la sua logica, è un flusso di byte ed è proprio sfruttando questo semplice assunto che saremo in grado di persistere da e verso un dataset e, quindi, da e verso i campi blob di un database.

Un esempio pratico: leggiamo un file da disco e da questo ricaviamo il vettore di byte corrispondente al suo contenuto binario:

```
//apriamo lo stream direttamente a partire dal file
System.IO.File file =System.IO.File.Open(
                              @"nomefile.bin", FileMode.Open);
// agganciamo un BinaryReader allo stream
// per semplificare le operazioni di lettura dei dati
System.IO.BinaryReader reader = new
                              System.IO.BinaryReader(file);
//copiamo il contenuto binario del file in un vettore
                                                   di byte
byte[] binArray = reader.ReadBytes((int) file.Length);
//rilasciamo il file
file.Close();
```

Una volta ottenuto il vettore di byte, lo si può adoperare per scopi vari, passando anche per la possibilità di persistere il vettore in un qualsiasi dispositivo di memorizzazione di dati… compreso il nostro caro database.

Quasi tutti i database, infatti, offrono un tipo di campo che, nella terminologia comune, è definito di tipo blob, cioè letteralmente una massa indistinta di bit nei quali mettere qualsiasi cosa. In alcuni database questo campo assume nomi meno orridi e più eleganti quali *image* o *binary*, ma il concetto non cambia. Ed è proprio sfruttando questa funzionalità che si rende possibile persistere in un database un qualsiasi file o flusso binario.

Naturalmente possiamo anche ricostituire un file o un qualsiasi stream binario a partire da una sequenza di byte che, guarda caso, recuperiamo proprio a partire dal database.

Ecco un esempio di codice:

```
byte[] data = <flusso di byte ottenuto da una lettura
              dal database o in qualsiasi altra forma>;
//istanziazione del memory stream
//che conterrà la riproduzione del flusso di byte
MemoryStream stream = new MemoryStream();

//istanziazione del BinaryWriter che si occuperà
```

```
//del riempimento dello stream a partire dai byte
BinaryWriter writer = new BinaryWriter(stream);

//scrittura dei byte
writer.Write(data);

//riposizionamento all'inizio dello stream in modo da
                        permetterne la fruizione completa
stream.Position = 0;
```

Ma cosa conterrà lo stream che abbiamo trasformato in un flusso di byte nel primo caso o, viceversa, che abbiamo ricostituito a partire da un flusso di byte?
In realtà potrà contenere qualsiasi cosa: da un banale file di testo, ad un'immagine, ad un mp3 o, semplicemente, l'istanza di una classe…

STRUTTURA DELLA TABELLA OSPITE

Allo scopo di ospitare i nostri nuovi fiammanti stream binari, predisponiamo una semplicissima tabella in un database Microsoft SQL Server. È evidente che con modifiche minime se non nulle è possibile fare altrettanto con qualsiasi altro tipo di database, sia di tipo client (i modesti *.dbf, Paradox* o *Microsoft Access*) che di tipo server (*DB2, Oracle, Sybase, Interbase*, ecc…).

SQL Server Enterprise Manager - [Design Table 'Blobs' ...

Column Name	Data Type	Length	Allow Nulls
IdBlob	int	4	
Description	varchar	255	
Data	image	16	☐
BlobType	int	4	

Fig. 1: **La struttura della tabella degli stream**

In **Figura 1** è possibile osservare la versione grafica della struttura della nostra tabella. È presente il campo chiave, *IdBlob*, che identifica in modo univoco lo stream, il campo *Description*, il campo *BlobType* che enumera le tre diverse tipologie di stream che andremo a gestire e infine il campo principale, *Data*, che sarà proprio quello che conterrà la sequenza di byte dello stream.
In SQL Server il campo blob assume il nome image, anche se il nome non deve trarre in inganno perché al suo interno potremo salvare qualsiasi cosa. Allo scopo di poter effettuare la lettura di uno stream dalla tabella, predisponiamo la semplicissima stored procedure di lettura *GetBlob*. Ad essa è sufficiente passare l'identificativo del blob. Per le operazioni di modifica ed inserimento, invece, ci affideremo alla stored procedure *SetBlob*. Essa viene utilizzata in modo duale: cioè controlla preventivamente che il dato da inserire non esista già nella tabella con una select; in caso affermativo si limita ad effettuare un update del record, diversamente ne esegue l'insert. Infine viene riletto e restituito per intero il record.
Ed ovviamente non può mancare nemmeno la stored procedure di cancellazione dello stream *(DeleteBlob)*. Queste stored procedure diventeranno rispettivamente i *CommandText* del command di lettura *(GetBlob)*, inserimento *(SetBlob)*, aggiornamento *(SetBlob)* e cancellazione *(DeleteBlob)*, operazione effettuata nel metodo *InitializeComponent* della classe *ResourceManager*.

LETTURA E SCRITTURA DEGLI STREAM SUL DATABASE

La classe *ResourceManager* che ci accingiamo a scrivere, disporrà di tutti i metodi per effettuare la lettura e la scrittura di stream binari verso e dalla tabella definita in precedenza. Necessiterà, come unica informazione, della *connectionstring* di connessione al database nel costruttore.
E così non ci resta che definire i metodi di scrittura e lettura degli stream. Partiamo dalla scrittura; supponiamo di voler persistere nella nostra tabella uno stream binario passato come parametro, indipendentemente dal suo contenuto:

```
public bool SetStream(int streamId,
                        System.IO.Stream stream)
  {
    //stream è proprio lo stream che si intende
    // salvare, streamId è l'identificativo che
    // assumerà nella tabella lo stream viene
    // riposizionato all'inizio
    stream.Position = 0;

    //l'adapter che contiene i command a cui
    //sono state assegnate le 4 stored procedere
    IDbDataAdapter da = daBlobs;

    //viene valorizzato il campo chiave del record
    IDbDataParameter param =
            da.InsertCommand.CreateParameter();
```

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

```
param.ParameterName= "@IdBlob";
param.DbType=DbType.Int32;
param.Value= streamId;
da.InsertCommand.Parameters.Add(param);

// il campo tipologia
param = da.InsertCommand.CreateParameter();
param.ParameterName= "@BlobType";
param.DbType=DbType.Int32;
param.Value= 1;
da.InsertCommand.Parameters.Add(param);

// una breve descrizione del contenuto
param = da.InsertCommand.CreateParameter();
param.ParameterName= "@Description";
param.DbType=DbType.String;
param.Value= "Stream";
da.InsertCommand.Parameters.Add(param);

// e il campo blob vero e proprio
param = da.InsertCommand.CreateParameter();
param.ParameterName= "@Data";
//si osservi che il tipo del campo viene definito
// come un generico Binary
param.DbType=DbType.Binary;
//viene istanziato un reader in grado di esplorare
// il contenuto dello stream da salvare
BinaryReader reader = new
                          BinaryReader(stream);

//si leggono i byte dallo stream e li si memorizzo
// in un array di byte
param.Value = reader.ReadBytes(3
                          (int)(stream.Length));
da.InsertCommand.Parameters.Add(param);
try
{
  da.InsertCommand.Connection.Open();

  //viene effettuata l'operazione di scrittura
                  eseguendo il command di insert
   da.InsertCommand.ExecuteNonQuery();
   da.InsertCommand.Connection.Close();
}
catch ( Exception ex )
{
  return false;
}
return true;
}
```

PERSISTERE UNA CLASSE

Il supporto alla persistenza di oggetti in .NET è pervasivo. Molte classi del framework sono dotate nativamente della capacità di persistere il proprio stato in modo da consentirne la serializzazione e l'eventuale trasferimento verso altri *AppDomain* che possano risiedere anche su macchine diverse.
È possibile inoltre aggiungere il supporto alla serializzazione anche nelle proprie classi, semplicemente implementando l'interfaccia *ISerializable* oppure adottando l'attributo *Serializable*.
Sfruttando questa capacità e abbinandola a quanto finora analizzato potremmo, ad esempio, persistere l'istanza di una nostra classe applicativa nel database in modo da poterla riesumare in qualsiasi momento, magari anche da una macchina che punti allo stesso database, e riottenendo così un'istanza identica a quella che era presente, nella macchina originaria, al momento del salvataggio originario. E questo senza doversi preoccupare di reimpostare tutte le proprietà interne della classe, ma semplicemente ripristinando l'istanza originaria.
La serializzazione delle classi in .NET è affidata ad un componente di sistema detto *Formatter.*
Come per tutte le implementazioni .NET, anche per il *Formatter Microsoft* non si è limitata soltanto a fornire delle librerie pronte all'uso, ma ha messo a punto un vero sistema astratto e estendibile di serializzazione di oggetti. Infatti è possibile scrivere i propri *Formatter* che dovranno seguire le regole di polimorfismo imposte dalle specifiche di base dei formatter.
Il framework fornisce già due implementazione:

1. **il *BinaryFormatter*, che usa una rappresentazione binaria per leggere e scrivere le informazioni delle istanze e pertanto è molto compatta,**
2. **e il SoapFormatter che invece si affida all'XML definito con la specifica SOAP e pertanto è meno compatto, ma sicuramente più standard e può essere trasferito e manipolato senza problemi visto che altro non è che normale testo.**

Osserviamo un semplice esempio di utilizzo del metodo per salvare nella nostra tabella l'intero contenuto di un file:

```
FileStream file = File.Open(<nome_file>,
                                FileMode.Open);
ResourceManager rm = new ResourceManager(
                            <connectionstring>);
rm.SetStream(1, file);
```

Pertanto, con sole tre istruzioni, siamo in grado di realizzare la magia: inserire un file in un database. Il metodo che realizza l'operazione contraria, *GetStream*, è persino più semplice. Eccolo:

```
public Stream GetStream(int streamId)
{
  DataTable dt;
  IDbDataAdapter da;
  IDbDataParameter param1;
  // daBlobs è il dataadapter membro della classe
  // inizializzato con il command di lettura che fa uso
  // della stored procedure GetBlob
  da = daBlobs;

  // valorizzazione del parametro IdBlob con lo
  //streamId passato al metodo
  param1= da.SelectCommand.CreateParameter();
  param1.ParameterName= "@IdBlob";
  param1.DbType=DbType.Int32;
  param1.11Value= streamId;
```

```
    da .SelectCommand.Parameters.Add(param1);

    //viene creato un dataset vuoto cha avrà lo scopo
    // di contenere il record ottenuto dalla select con il
    // metodo Fill() del dataadapter
    DataSet ds = new DataSet();
    da.Fill(ds);
    dt = ds.Tables[0];
    if ( dt.Rows.Count == 0 ) return null;

    //viene creato un memory stream che verrà
    // riempito col contenuto del blob del record letto
    MemoryStream stream = new MemoryStream();

    //viene creato un binary writer che si
    // incaricherà di scrivere nel memory strem
    BinaryWriter writer = new BinaryWriter(stream);

    //il contenuto del campo Data della tabella, che è
    // finito nel dataset dopo la lettura, viene
    // semplicemente scritto nel memory stream
    // grazie al binary writer
    writer.Write((byte[]) dt.Rows[0]["Data"]);

    //lo stream viene riposizionato all'inizio
    stream.Position = 0;

    //ed eccolo pronto ad essere restituito al
    // chiamante…
    return stream;
}
```

Il codice seguente, apparentemente complesso, non fa altro che richiedere alla nostra classe uno stream a partire da un id di blob presente nella tabella e procedere al suo salvataggio su file per una più semplice fruizione:

```
ResourceManager rm = new ResourceManager(
                              <connectionstring>);
//la nostra classe restituisce lo stream contenuto
// nel record con chiave 1
Stream stream = rm.GetStream(1);
```

Sono pertanto bastate due istruzioni per ottenere uno stream a partire dal flusso di dati salvato nel record. Se volessimo leggerne il contenuto, potremmo semplicemente salvare lo stream su file inserendolo su un *FileStream*:

```
FileStream fs = File.OpenWrite(
                          <nome_file_da_salvare>);
BinaryWriter writer = new BinaryWriter(fs);
BinaryReader reader = new BinaryReader(stream);
writer.Write(reader.ReadBytes((int) stream.Length));
stream.Close();
fs.Close();
```

PERSISTENZA DI IMMAGINI

Abbiamo più volte ribadito, che ogni cosa in informatica è un flusso di byte, i nostri stream potranno contenere qualsiasi cosa, sarà a cura dell'utilizzatore della nostra semplice ma preziosa classe *ResourceManager* decidere cosa conterrà il flusso e come dovrà essere trattato.
Ma perché non provare a fornire una sorta di specializzazione dei nostri metodi per leggere e scrivere direttamente immagini (intese come figura, fotografie, disegni)?
Allo scopo ci viene incontro il sempre efficiente .NET Framework, attraverso la classe *System .Drawing.Image*, che fornisce metodi per creare immagini a partire da stream e viceversa.
Ecco un esempio per salvare un'immagine in uno stream:

```
//carichiamo in memoria un'immagine (.bmp, .jpg, o
//qualsiasi altro dei numerosi formati supportati)
Image image = Image.FromFile(
                           <path_della_nostra_immagine>);

//istanziamo il memory stream che conterrà la
// versione serializzata della nostra immagine
MemoryStream stream = new MemoryStream();

//invochiamo il metodo Save di Image a cui
// passiamo lo stream nel quale verrà serializzata
// l'immagine stessa nel formato indicato dal
// secondo parametro
image.Save(stream,
          System.Drawing.Imaging.ImageFormat.Jpeg);

// riposizioniamo lo stream all'inizio che sarà
// contenuto così pronto all'uso
stream.Position = 0;
```

A questo punto siamo in grado di persistere la nostra immagine nella nostra solita tabella nel database passando semplicemente lo stream ottenuto.
L'operazione contraria sarà altrettanto semplice:

```
//creazione del memory stream che conterrà il flusso
//dell'immagine a partire dal file dell'immagine
FileStream stream = File.Open(<nome_file>,
                                      FileMode.Open);
stream.Position = 0;
//caricamento dell'immagine a partire dallo stream
Image image = Image.FromStream(stream);
```

È evidente che con poche modifiche è possibile realizzare due versioni specializzate delle

DOVE SONO LE STORED PROCEDURE?

Le stored procedure utilizzate all'interno di questo articolo sono riportate nel file *storedprocedure.txt* contenuto nel codice allegato

precedenti *GetStream* e *SetStream* specifiche per la gestione delle immagini.
Ed ecco i metodi *GetImage* e *SetImage* di cui si rimanda la consultazione del codice sorgente allegato.
In **Figura 2** è possibile osservare il client *Winforms* di esempio che sfrutta il *ResourceManager* per leggere e scrivere stream e immagini.

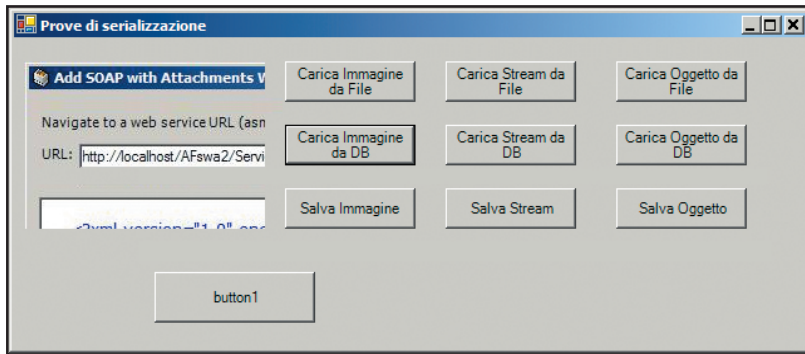



Fig. 2: ***Il client di esempio***

SERIALIZZARE ISTANZE DI OGGETTI NEL DATABASE

A questo punto non ci resta che realizzare una semplice funzione in grado di persistere istanze di oggetti in uno Stream:

```
public enum FormatterType
{
    Binary,
    SOAP
}

private Stream saveObjectState(object obj,
                               FormatterType formatterType)
{
    Stream stream = new System.IO.MemoryStream();
    IFormatter formatter;
    if (formatterType == FormatterType.Binary)
        formatter = new BinaryFormatter();
    else
        formatter = new SoapFormatter();
        // Serializzazione dello stato dell'istanza
       formatter.Serialize(stream, obj);
    return stream;
}
```

Ed ecco un esempio:

```
//istanziazione di un hashtable, classe
//serializzabile by design
Hashtable coll = new Hashtable();

//nell'hashtable vengono caricati dei valori
//scalari (un numero e una stringa)
coll["uno"] = 45;
coll["due"] = "prova";

//e un intero datatable costituito da
//due campi e già riempito con un record
DataTable dt = new DataTable();
dt.Columns.Add("Codice", typeof(int));
dt.Columns.Add("Descrizione", typeof(string));
DataRow row = dt.NewRow();
row["Codice"] = 234;
row["Descrizione"] = "Codice 234";
dt.Rows.Add(row);
coll["tre"] = dt;

// l'hashtable così ottenuto viene passato alla
// funzione vista in precedenza che restituisce
// uno stream che contiene la serializzazione
// dell'istanza dell'hashtable passato
Stream stream = saveObjectState (
                    coll, FormatterType.Binary);
```

Lo stream così ottenuto può essere dato in pasto alla nostra tabella *SQL Server* che così potrà ospitare, oltre a generici stream o immagini, anche intere istanze delle nostre classi. Infatti *ResourceManager* presenta due metodi specifici per serializzare istanze di oggetti su database *(SetObject)* e deserializzarle da database *(GetObject)*.
La deserializzazione di un'istanza a partire da uno stream è un'operazione altrettanto semplice, come si può osservare dalla seguente funzione:

```
object restoreObjectState(Stream stream,
                          FormatterType formatterType)
{
    IFormatter formatter;
    if (formatterType == FormatterType.Binary)
        formatter = new BinaryFormatter();
    else
        formatter = new SoapFormatter();
        stream.Position = 0;

        // deserializzazione dell'istanza dallo stream
    return formatter.Deserialize(stream);
}
```

Ed ecco un esempio di deserializzazione del nostro hashtable visto in precedenza da uno stream usando la funzione *restoreObjectState:*

```
MemoryStream stream = <caricamento dello stream
                       contenente l'istanza>;
Hashtable coll = (Hashtable) restoreObjectState(
                    stream, FormatterType.Binary);
```

IL PASSAGGIO FINALE: LA CLASS ADOSTREAM

Il viaggio nel mondo degli stream e del loro comodo adattamento nell'apparentemente lontano mondo dei database non può che portare all'approdo finale: la realizzazione di una classe derivata *Stream* che gestisca nativamente la lettura e la scrittura di flussi da e verso un database relazionale.
Allo scopo di semplificare la presentazione, si assumerà che questo Stream opererà sulla tabella *Blobs* finora utilizzata nelle prove sfruttando le stored procedure già viste. Infatti internamente lo stream non farà altro che utilizzare il *ResourceManager* per effettuare tutte le operazioni.
Ed eccola:

```
// la classe eredita da MemoryStream in modo
// da avere già un'implementazione ricca e
// di tutto ciò che serve ad uno Stream secondo
// consistente le specifiche del .NET Framework
// la classe verrà semplicemente arricchita
// con il supporto alla serializzazione su database
public class ADOStream : MemoryStream
{
  // istanza interna del ResourceManager
  ResourceManager _rm;
  int _streamId;
  // al costruttore dello stream verrà
  // passata la connectionstring di connessione
  // al database che conterrà la tabella Blobs
  // e l'identificativo della risorsa su
  // cui lo stream dovrà agire, in pratica
  // il record di Blobs cu cui dovrà agire
  // se il record non esiste, verrà
  // generato un nuovo record con l'id indicato
  public ADOStream(string connectionstring, int
                                       streamId )
  {
     // istanziazione del ResourceManager interno
     _rm = new ResourceManager(connectionstring);
     _streamId = streamId;
     Stream thisStream = this;
     // precaricamento dello stream a
     // partire dal record indicato
     // dallo streamId, se questo è già
     // presente nella tabella
     _rm._GetStream(_streamId, ref thisStream);
  }
  // l'unico metodo che viene ridefinito è
  // Close, perché solo in fase di chiusura
  // interverrà l'unica differenza rispetto ad un
  // normale MemoryStream, infatti in questo
  // caso il contenuto dello stream verrà
  // direttamente serializzato su db sfruttando
  // il ResourceManager interno
  public override void Close()
  {
     if ( _rm == null ) throw new Exception("La
      connessione al database non è stata effettuta");
     _rm.SetStream( _streamId, this );
     base.Close();
  }
}
```

Ed ecco un esempio significativo di uso del nostro nuovo *ADOStream*:

```
//istanziazione di un ADOStream con id = 4
ADOStream stream = new
                ADOStream(<connectionstring>, 4);
//apertura di un file che verrà poi riversato
//nel nostro ADOStream
FileStream file = File.Open(@"c:\prova.doc",
                                   FileMode.Open);
BinaryReader reader = new BinaryReader(file);
BinaryWriter writer = new BinaryWriter(stream);
writer.Write(reader.ReadBytes((int) file.Length));
file.Close();
// l'invocazione del metodo Close farà avvenire
// l'effettiva scrittura dell'ADOStream su database
stream.Close();

// operazione all'inverso: l'ADOStream viene usato
// per recuperare uno stream da database
stream = new ADOStream(<connectionstring>, 4);
FileStream fs = File.OpenWrite(@"c:\prova1.doc");
writer = new BinaryWriter(fs);
reader = new BinaryReader(stream);
writer.Write(reader.ReadBytes((int) stream.Length));
stream.Close();
fs.Close();
```

CONCLUSIONI

Lo streaming di dati su database è un altro esempio di come combinare varie tecnologie già presenti nel framework per ottenere soluzioni completamente nuove e che possano essere utilizzate in contesti applicative reali.
Il progetto completo mostrato nel corso dell'articolo rappresenta una soluzione estensiva dei concetti esposti, ma potrà essere certamente migliorata rendendo dinamica e configurabile la scelta della tabella in cui persistere i dati o fornendo alla classe *ADOStream facility* più avanzate quali il supporto a tipologie di dati più specifici come già avviene nel *ResourceManager* sottostante.
Ma naturalmente queste ed altre soluzioni potranno essere il terreno su cui confrontarvi se deciderete di approfondirne lo studio in vista di un utilizzo reale.

Vito Vessia

AUTORI

Vito Vessia è cofondatore della codeBehind S.r.l. http://www.codeBehind.it, una software factory di applicazioni enterprise, web e mobile, dove progetta e sviluppa applicazioni e framework in .NET, COM(+) e Delphi occupandosi degli aspetti architetturali. È autore del libro "Programmare il cellulare", Hoepli, 2002, sulla programmazione dei telefoni cellulari connessi al PC con protocollo standard AT+. Può essere contattato tramite e-mail all'indirizzo vvessia@katamail.com.

DB Prevalence, la nuova frontiera

Muoviamo i primi passi nella tecnica che tutti i futuri DB stanno sperimentando, è più veloce, più stabile e più comoda da gestire. Cosa ci dovremo aspettare in futuro dai Database?

Uno sviluppatore prima o poi si trova a dover studiare un modo per consentire all'utente il salvataggio, ed il relativo recupero del lavoro che sta compiendo attraverso l'applicazione che ha realizzato. Sembra un problema abbastanza facile da risolvere a parole, vero? In teoria basterebbe stabilire un formato con il quale memorizzare lo stato e i dati del programma, realizzando una sorta di fotografia dell'istante di esecuzione, e sviluppare un modulo che si occupi del salvataggio di questi, ad esempio, su file. Lo stesso modulo dovrebbe anche essere in grado di ripristinare i dati contenuti nel file di salvataggio riportando l'applicazione al momento della "fotografia" scattata. In sostanza, il programmatore non deve fare altro che progettare il formato per il salvataggio dei dati e sviluppare le funzioni necessarie a implementarlo.

In un'applicazione realizzata mediante un linguaggio di programmazione Object Oriented, i dati "strutturati" (come ad esempio tutte le informazioni relative all'anagrafica di un cliente) sono rappresentati come oggetti: il voler salvare i dati sotto forma di oggetti e di conseguenza ripristinarli come tali in un secondo momento non è un problema di poco conto. Infatti, mediante un oggetto possiamo rappresentare logicamente qualsiasi informazione mediante l'indicazione di alcuni dati o proprietà (le variabili) e i metodi necessari a creare e manipolare l'informazione stessa. Ma le proprietà possono essere a loro volta altri oggetti. È possibile che uno stesso oggetto, inteso come singola istanza di una classe, sia contemporaneamente "condiviso" da altri, cioè costituisca una parte dell'informazione di altri oggetti (due variabili si riferiscono allo stesso oggetto). Ad esempio, due persone condividono lo stesso contocorrente bancario; gli oggetti in gioco sono due istanze di un'ipotetica classe *Persona* e un'istanza di *Conto*. Se supponiamo che la classe Persona abbia una variabile *cc* di tipo *Conto*, affinché due oggetti di tipo *Persona* si riferiscano alla stessa istanza di *Conto*, entrambe le variabili *cc* dei due oggetti *Persona* dovranno riferirsi al medesimo oggetto *Conto*. Se registrassimo un bonifico su questo conto, entrambe le persone dovranno avere traccia di questa operazione. Archiviare un oggetto e ripristinarlo vuol dire quindi mantenere anche questi riferimenti incrociati. Al salvataggio di un oggetto *Persona* dobbiamo tenere traccia anche del relativo oggetto *Conto;* di conseguenza, al ripristino di una *Persona* è necessario recuperare anche il relativo *Conto*.

Per questo motivo l'archiviazione dei dati in un programma Object Oriented non è un problema semplice da affrontare.

COME INIZIARE

È necessario scaricare le librerie per implementare la persistenza tramite la prevalenza all'indirizzo http://www.prevayler.org. Qui si possono trovare oltre alle ultime librerie jar necessarie anche tantissimo materiale e articoli di riferimento sull'Object Prevalence in generale. Nello stesso sito trovate anche Bamboo per Dot.Neta

OBJECT PERSISTENCE

La capacità di un oggetto di esistere oltre l'esecuzione del programma che lo ha generato è nota come persistenza. La chiave dell'implementazione della persistenza è la serializzazione che offre la la capacità di scrivere un oggetto su un flusso di dati (ad esempio uno stream che opera su di un file)

```
Hashttable objectH=....
BinaryFormatter  formatter = new BinaryFormatter();
FileStream fstream = new FileStream("DataFile.dat ",
FileMode.Create);
formatter.Serialize(fstream, objectH);
fstream.Close();
```

e di rileggerlo in un secondo momento

```
BinaryFormatter formatter = new BinaryFormatter();
FileStream fstream = new FileStream("DataFile.dat",
FileMode.Open);
Hashtable objectH = (Hashtable)
formatter.Deserialize(fstream);
```

Linguaggi di programmazione come Java e C# offrono ai programmatori dei meccanismi per realizzare la persistenza. In sostanza la serializzazione consente di archiviare gli oggetti su file, spostarli attraverso una rete di computer interconnessi (come internet!) e di trasferirli da un'applicazione ad un'altra che sta operando su di una macchina diversa. Nella serializzazione di oggetti, oltre alla memorizzazione delle informazioni in esse rappresentate si salvano anche quelle relative alla classe da cui sono istanziate. Senza le informazioni sulla classe non esisterebbe il modo di ricostruire un oggetto serializzato. Inoltre, è necessario tenere traccia anche tutti gli oggetti a cui l'oggetto in questione fa riferimento: altrimenti, se questi non fossero a loro volta archiviati, i vari collegamenti sarebbero privi di significato perché non corrisponderebbero ad istanze reali effettivamente presenti.

LIMITI DELLA SERIALIZZAZIONE

Di solito, il procedimento di serializzazione fa uso di uno stream: gli oggetti vengono salvati e scritti su questo stream per poi essere successivamente ripristinati rispettando però la sequenza di scrittura. È quindi evidente che per accedere ad un singolo oggetto è necessario ripristinare l'intera sequenza o per lo meno ignorare tutti quelli precedenti a quello cercato. Non esiste altro modo per "interrogare" l'archivio alla ricerca di un data informazione se non quello di scandirlo record per record.
Spesso, per risolvere i problemi di ricerca, il programmatore tende ad usare un database relazionale per l'archiviazione, scomponendo i dati in elementi logici messi in relazione tra loro: ciascun oggetto viene memorizzato nel database sotto forma di record e gli eventuali riferimenti ad altri oggetti tramite un opportuno valore in un campo. In questo modo, tutte le operazioni di inserimento e di ricerca degli oggetti sono delegati al RDBMS *(Relational DataBase Management System)* impiegato tramite un interfacciamento che utilizza il linguaggio di interrogazione SQL. È necessario quindi modellare opportunamente le classi in base a come i dati sono conservati nel database e scrivere le opportune query SQL da utilizzare per recuperare o modificare i dati: le operazioni che devono essere effettuate sui dati vengono demandate quindi alle applicazioni che li utilizzano. Questa tecnica "distrugge" l'oggetto, smembrandolo: prima di essere reinstanziato all'interno dell'applicazione è necessario riassemblarlo pezzo per pezzo, compresi i vari riferimenti ad altri oggetti. Esiste una possibilità che consente di adottare sempre come archivio dei dati un database, non più come semplice contenitore di informazioni ma di veri e propri oggetti: gli OODBMS *(Object-oriented Database Management System)*. I database ad oggetti possono in effetti essere visti come l'evoluzione dei database relazionali; infatti, ad essi è stata data la possibilità di definire nuovi tipi di dati e comportamenti, che li rende molto utili a contenere oggetti come li abbiamo appena definiti e quindi utili a supportare il meccanismo di serializzazione introdotto in un'applicazione. Purtroppo non esiste un modello di oggetto universalmente accettato e, di conseguenza, un linguaggio di interrogazione standard come l'SQL; ciò comporta che ogni produttore può implementare la propria versione di OODBMS, di solito assolutamente incompatibile con quella dei concorrenti. Si stanno diffondendo alcuni database definiti *object-relational*, che cercano di introdurre nel modello relazionale le caratteristiche di estendibilità proprie dei database object-oriented.
L'utilizzo di un database per effettuare l'archiviazione dei dati attraverso la serializzazione introduce quindi una maggire complessità allo sviluppo dell'applicazione. Esiste una soluzione al problema della persistenza, che utilizza il meglio di ciascuna delle soluzioni appena descritte: l'*Object Prevalence*.

OBJECT PREVALENCE

Object prevalence è un concetto ideato da Klaus Wuestefeld e implementato la prima volta nel novembre del 2001 per Java con un progetto opensource chiamato *Prevayler* (*http://www.prevayler.org*). L'idea dietro a questo progetto è sostanzialmente quella di utilizzare i meccanismi della persistenza dei dati senza l'uso di un RDBMS, cioè attraverso oggetti e non tramite un modello di dati relazionale, sviluppando quindi il solo objectmodel. L'*Object Prevalence* è concettualmente molto semplice: può essere realizzato in qualsiasi linguaggio di programmazione ad oggetti che supporta la persistenza.
Vediamo come funziona il sistema (**Figura 1**).
In memoria vengono tenuti tutti i dati necessari all'applicazione attraverso istanze di oggetti (i *Business Object*). I comandi che effettuano una manipolazione di questi oggetti vengono serializzati e scritti in un apposito file di log, dopo di che possono essere eseguiti immediatamente.

I TUOI APPUNTI

BAMBOO

Esistono diversi porting del sistema Prevayler verso altre piattaforme diverse da Java. Ad esempio sotto Dot.Net abbiamo Bamboo, un progetto sempre open-source con licenza GPL (http://bbooprevalence.sourceforge.net). Il meccanismo alla base di tutte è sempre uguale, viene soltanto adattato ai costrutti del linguaggio di programmazione che si vuole usare. Per piattaforme semplici da configurare come lo sono Java e Dot.Net, basta soltanto portarsi dietro la relativa libreria per quell'ambiente e definirne il riferimento. In Java è necessario soltanto scaricare lo jar di Prevayler dal sito web http://www.prevayler.org e assicurarsi di averlo posizionato correttamente nel classpath che si andrà ad utilizzare. Per C#, nella versione di Bamboo, è necessario procurarsi il file *Bamboo.Prevalence.dll* dalla distribuzione disponibile sul sito e, se si utilizza Visual Studio.Net 2003, aggiungerlo al reference del progetto.

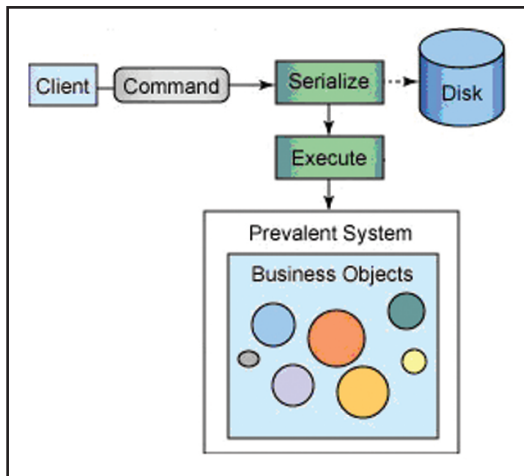



Fig. 1: ***Schema del modello adottato in un contesto di Object Prevalence.***

Quando il sistema è relativamente scarico, si può effettuare una "istantanea" del suo stato, una sorta di fotografia della memoria dell'applicazione, originando un file di tutti i business object e aggregando in un unico file anche tutti i comandi effettuati; il file di snapshot (così è identificato questo file di immagine) viene salvato per rendere disponibile lo stato dell'applicazione per un recupero successivo. Ad ogni riavvio del sistema, se è disponibile, viene recuperato lo stato dall'ultimo file di snapshot e si procede alla lettura dei comandi registrati nei file di log fino al momento in cui è stato generato lo snapshot. Una volta recuperati questi comandi, essi vengono poi eseguiti sui business object come se essi fossero appena richiesti dall'applicazione.
Così, il sistema ritorna allo stato in cui si trovava subito prima della sua chiusura ed è di nuovo pronto a continuare. Questo risulta molto utile anche nel caso di crash accidentale del sistema.Per poter funzionare correttamente, gli oggetti con cui vengono rappresentate le informazioni, i *business object* appunto, in un sistema basato sull'*Object Prevalence* devono soddisfare due semplici condizioni:"

- **La serializzazione:** ad ogni istante durante l'esecuzione, il sistema può salvare su file l'oggetto."

- **Il determinismo:** Dato lo stesso input, I metodi richiamati sui business object devono sempre ritornare il medesimo output. In questo modo non è necessario utilizzare un RDBMS poiché tutti gli oggetti sono mantenuti in memoria.
  Ciò permette inoltre un accesso alle informazioni estremamente veloce e rende possibile mantenerle sempre rappresentate mediante un modello object-oriented.

IN PRATICA

Mostriamo adesso come sia veramente facile adottare un sistema *Object Prevalence*. Costruiamo un programma in C# che memorizzi un elenco di utenti e che ci consenta di mantenerli sempre aggiornarti. Per semplicità, prevediamo solo la possibilità di aggiungere un nuovo utente e di visualizzare i dati di ciascuno (**Figura 2**).

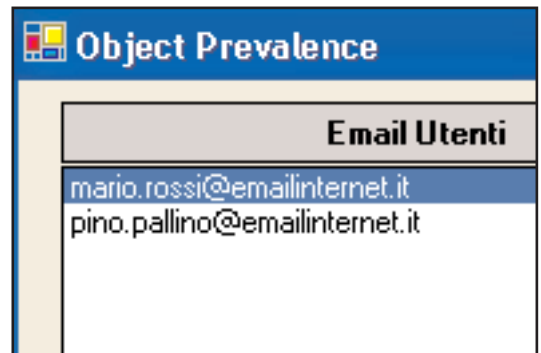



Fig. 2: ***La finestra principale dell'applicazione. Il pulsante SnapShot servirà per creare un'istantanea dello stato dell'applicazione.***

Definiamo una classe *User* con cui istanziare semplicemente gli oggetti utenti come serializzabili:

```
[Serializable]
public class User
{
 private string mName;
 private string mEmailAddress;
 public string Name{get{return mName;}}
 public string EmailAddress{get{return
mEmailAddress;}}
 public User(string name, string emailAddress)
 {
   mName = name;
   mEmailAddress = emailAddress;
 }
}
```

E definiamo un gestore, l'*UserManager*, che definisca l'*Object Prevalence*

```
[Serializable]
public sealed class UserManager :
MarshalByReObject
{
  #region Prevayler implementation
   ...
  #endregion
  public event EventHandler CmdExecuted;
```

```
  private Hashtable mUsers = new Hashtable();
  [Query]
  public Hashtable GetAllUsers()...
  public ArrayList GetUsers()...
  public void setUsers(Hashtable users)...
  [PassThrough]
  public Object ExecuteCommand(ICommand
command)
  {
      Object o=
mEngine.ExecuteCommand(command);
      CmdExecuted(this, null);
      return o;
  }
  public void TakeSnapshot()
  {        mEngine.TakeSnapshot();
  }
}
```

Ogni qual volta vogliamo aggiungere un utente alla lista, costruiamo un'istanza del comando e lo mandiamo in esecuzione sull'istanza comune del *PrevalentSystem*

```
...
string name = txtName.Text;
string emailAddress = txtEmailAddress.Text;
User utente=new User(name, emailAddress);
AddUser cmd = new AddUser();
cmd.Utente=utente;
try
{
  UserManager.Instance.ExecuteCommand(cmd);
}
catch (Exception ex)
{
  MessageBox.Show("Error: " + ex.Message);
}
...
```

Vediamo la region *"Prevayler implementation"* nella quale aggianceremo il sistema

```
#region Singleton and prevayler implementation
static private PrevalenceEngine mEngine;
static public UserManager Instance=new
UserManager();
static UserManager()
{ string path = Path.Combine(
                         Environment.CurrentDirectory,
                                              "Data");
  mEngine =
            PrevalenceActivator.CreateTransparentEngine(
                                   typeof(UserManager),
                                                 path);
  Instance = mEngine.PrevalentSystem as
UserManager;
}
```

```
public UserManager(){}
#endregion
```

Con *CreateTrasparentEngine* otteniamo il motore del *Prevalence System*, inserendovi il sistema di gestione da noi definito (l'*UserManager*) ed ottenendo l'istanza del gestore che utilizzeremo sia per recuperare i dati (ciascun oggetto *User*) che per eseguire ogni operazione di inserimento.
Il meccanismo può sembrare un po' artificioso in quanto dobbiamo sempre passare attraverso la costruzione esplicita di un comando *AddUser* per aggiungere un oggetto *User* alla lista, mentre per recuperare l'elenco dei record possiamo richiederlo direttamente al gestore. Questo perché abbiamo rispettato il protocollo previsto dal *Prevayler* che prevede la generazione di espliciti comandi da parte del client che accede al sistema.
Possiamo modificare leggermente l'operazione di inserimento in modo da non dover far ricorso esplicitamente alla formulazione di questi comandi. Nel gestore definiamo un nuovo metodo, il quale banalmente aggiunge all'hashtable l'istanza dell'oggetto *User* solo se esso è valido (nome ed email non nulli o vuoti ed email non ancora presente)

```
...
public void AddUser(User user)
{
 if (user == null) throw new Exception("null user");
 if (user.Name == null || user.Name == string.Empty)
             throw new Exception("Invalid Name");
 if (user.EmailAddress == null ||
                         user.EmailAddress == string.Empty)
             throw new Exception("Invalid Email");
if (mUsers.ContainsKey(user.EmailAddress))
      throw new Exception("User already exists");
mUsers.Add(user.EmailAddress, user);
...
}
...
```

in questo modo, ogni qual volta che si dovrà inserire un nuovo oggetto all'archivio, basterà aggiungerlo direttamente nell'hashtable del gestore.
Dall'esterno basterà richiamare soltanto il metodo *AddUser*

```
...
User utente=new User(name, emailAddress);
try
{
  UserManager.Instance.AddUser(utente);
}
catch (Exception ex)
{
  MessageBox.Show("Error: " + ex.Message);
```

```
}
...
```

ASP.NET

La poliedricità del sistema ci consente di sviluppare qualsiasi tipo di applicazione che sfrutta l'Object Prevalence per realizzare la persistenza dei dati (e dello stato) in gioco, senza passare attraverso un database. Nulla ci vieta di adottare il meccanismo anche per un'applicazione WEB! Spesso riusciamo a trovare con poca spesa dei provider che offrono dello spazio Web con il supporto alle applicazioni Asp.Net; queste offerte non comprendono l'utilizzo di un database.
L'Object Prevalence ci viene in aiuto per soddisfare in maniera molto agevole questa necessità.
Supponiamo di voler portare su internet il nostro elenco degli utenti.
Per fare ciò basterà modificare soltanto il metodo statico dell'*UserManager* che ci fornisce l'istanza del sistema *Prevalence*

```
static UserManager()
{
string path = HttpContext.Current.Server.MapPath(".");
mEngine =
          PrevalenceActivator.CreateTransparentEngine(
                              typeof(UserManager), path);
Instance = mEngine.PrevalentSystem as UserManager;
SnapshotTaker taker = new
        SnapshotTaker(mEngine,TimeSpan.FromHours(1),
   Bamboo.Prevalence.Util.CleanUpAllFilesPolicy.Default);
}
```

Se lo mettiamo a confronto con l'altra versione, si nota subito che viene indicato in maniera diversa il percorso che definisce la cartella in cui inserire i file di log e i file snapshot. In più, l'operazione di salvataggio dello stato dell'applicazione (in modo da avere un ripristino in caso di crash) viene delegata, tramite lo *SnapshotTaker,* all'engine del sistema che lo eseguirà automaticamente ad ogni ora.
Per il resto, la gestione rimane praticamente inalterata. Basterà quindi effettuare il porting delle windows forms verso pagine aspx.
Naturalmente se i dati in gioco sono troppo numerosi questa soluzione diventa un po' pesante da gestire, in quanto andrebbero tenuti in memoria anche quei dati non più richiesti perché troppo vecchi o di scarso interesse (ad esempio, per un sito di e-commerce i dati superflui potrebbero essere gli ordini dell'anno precedente).

CONCLUSIONI

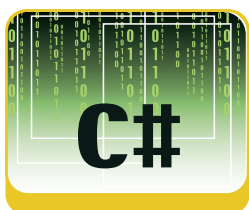
L'Object Prevalence a giusta ragione può essere considerata l'evoluzione della serializzazione, perchè va oltre al semplice concetto di salvataggio su stream dei dati. Tramite questo meccanismo è possibile mantenere sempre serializzata l'intera memoria dell'applicazione e, a richiesta, effettuare un'istantanea anche dello stato stesso in modo da consentirne il ripristino in qualsiasi momento, continuando l'esecuzione come se non fosse mai stata terminata. Questo è certo e abbiamo anche visto come realizzarlo con un semplice esempio. Ma in termini di prestazioni cosa possiamo dire? Beh, da un confronto tra un'applicazione scritta in Java utilizzando *Prevayler* e la stessa che si appoggia ad un database relazionale (e quindi che fa uso di JDBC), le interrogazioni in *Prevayler* fatte utilizzando collection di dati e algoritmi specifici sono circa 9000 volte più veloci di quelle fatte utilizzando query per Oracle e circa 3000 volte più veloci di quelle per MySQL (fonti *http://www.prevayler.org*).
Un bel risultato, non c'è che dire!

Ing. Antonino Panella

DATABASE OBJECT-ORIENTED

**I database appartenenti a questa categoria si basano sul modello relazionale la cui struttura principale e' la relazione, cioè una tabella bidimensionale composta da righe e colonne.
Ciascuna riga, che in terminologia relazionale viene chiamata tupla, rappresenta un'entità che noi vogliamo memorizzare nel database. Le caratteristiche di ciascuna entità sono definite invece dalle colonne delle relazioni, che vengono chiamate attributi. Entita' con caratteristiche comuni, cioè descritti dallo stesso insieme di attributi, faranno parte della stessa relazione.
lo schema di un database ad oggetti e' rappresentato da un insieme di classi, che definiscono le caratteristiche ed il comportamento degli oggetti che popoleranno il database. Questi oggetti contengono sia i dati che le operazioni possibili su tali dati.
In un certo senso potremmo pensare agli oggetti come a dati dotati di una certa indipendenza, che gli permette di sapere come comportarsi in conseguenza di specifiche richieste, senza doversi appoggiare ad applicazioni esterne.**

NewsReader? Facciamolo con C#!

I newsgroup sono una delle maggiori risorse che la rete offre al suo vasto pubblico. Impariamo come funziona il protocollo per inviare e ricevere messaggi e creiamo un lettore di news

L'acronimo NNTP sta per *Net News Transfer Protocol*. NNTP è il protocollo utilizzato per la lettura e scrittura di messaggi su newsgroup, per la memorizzazione dei messaggi, del loro trasferimento fra diversi server, e di altre utilità necessarie al funzionamenti di USENET.
È un protocollo di tipo *request-reply*, sulla tipologia ad esempio di FTP e SMTP, basato quindi sullo scambio di messaggi testuali, che usano determinate sintassi e convenzioni, e che viaggiano su una connessione TCP bidirezionale. Il protocollo NNTP utilizza un trasporto TCP, e di default la porta utilizzata è la 119. Se avete a disposizione un server di news, ad esempio uno dei tanti gratuiti che si trovano con una ricerca su internet, potete anche provare a connettervi ad esso, ed inviare comandi, tramite una semplice connessione telnet. Nel protocollo NNTP e nelle relative estensioni, si usano comandi definiti come stringhe di caratteri ASCII, seguiti eventualmente da uno o più parametri. I comandi non sono case-sensitive, ed ogni linea di comando deve essere terminata da una coppia CR-LF *(Carriage Return - Line Feed)*. Inoltre, ogni linea di comando, deve essere al massimo di lunghezza 512 caratteri, compresi CR-LF, quindi per il comando e i parametri sono disponibili 510 caratteri. Le risposte ad ogni comando inviato ad un server sono invece di due tipi: di stato e testuali. Le risposte di stato indicano la risposta del server ad un comando inviato da un client.
Iniziano con un codice numerico di 3 cifre, che è già sufficiente per comprendere la risposta.

STORIA DI UNA CONNESSIONE

Cominciamo a fare pratica con una connessione telnet ad un server di news, in modo da conoscere i comandi da implementare e le risposte che ci si può attendere dal server. Lanciate il comando seguente, naturalmente con una connessione di rete attiva, e con un server valido:

```
telnet nomeserver 119
```

Se la connessione va a buon fine, il server risponderà con qualcosa del genere:

```
200 Welcome to News server
```

In questo caso il server ha risposto con il codice di stato 200, seguito dal nome del server stesso.
Il codice 200 indica che il server è pronto, ed è consentito spedire nuovi messaggi. Il 201 avrebbe invece indicato che il server sarebbe stato disponibile solo in modalità di lettura.
Ogni comando naturalmente avrà diversi possibili codici di risposta e li incontreremo man mano che andremo avanti nell'esposizione del protocollo.
Se è necessaria l'autenticazione al server, cosa che di solito il server indica nella risposta alla prima connessione, username e password verranno specificati con due comandi differenti ma complementari:

```
AUTHINFO USER utente
```

Se la coppia utente e password è corretta, il server risponderà a questo punto con il codice 281 di autenticazione corretta, altrimenti restituirà il codice di permesso negato 502.
Una volta autenticati, è possibile iniziare ad impartire comandi che riguardano specificatamente newsgroup e messaggi.
Questi comandi, oltre al codice di stato, riceveranno dal server anche una risposta testuale. Ad esempio è possibile richiedere al server l'elenco dei newsgroup tramite il comando *LIST*.
Tale elenco sarà restituito al client in una serie di

linee di testo, ognuna terminata da *CR-LF,* mentre la fine dell'elenco, ed in genere di ogni risposta testuale, sarà indicata da una linea di testo contenente un solo punto. Provate ad esempio ad inviare il comando *LIST* ancora tramite telnet.
Notate che, se la risposta è positiva, viene prima restituito il codice 215, ad indicare che seguirà un elenco di newsgroups, e quindi viene stampato un elenco di righe, contenenti nome del gruppo, il numero identificativo dell'ultimo messaggio e quello del primo, ed una *y* che indica il permesso di inviare messaggi al gruppo. A questo punto possiamo selezionare il gruppo mediante il comando *GROUP* seguito dal nome del newsgroup, ad esempio:

```
GROUP public.test
```

Se il newsgroup esiste, il server risponderà con il codice 211 e dei parametri che identificano rispettivamente il numero totale di messaggi, il primo e l'ultimo messaggio disponibile nel gruppo stesso.
Un messaggio, come vedremo meglio fra poco, è formato da un'intestazione e dal corpo che contiene il testo vero e proprio. Se vogliamo leggere l'intero messaggio, comprensivo di intestazione e corpo, si può utilizzare il comando *ARTICLE,* seguito dal numero del messaggio. Se invece ci interessa solo l'*header* o solo il *body,* sono disponibili i comandi *HEAD* e *BODY* rispettivamente. L'invio di un messaggio, confezionato come si deve, cioè completo ancora una volta di *header* e *body,* si effettua invece con il comando *POST.* Se il server è pronto a ricevere un messaggio, risponderà con il codice 340, che indica al client di iniziare la spedizione del messaggio. Il messaggio deve essere a questo punto spedito linea per linea, a partire da quelle che costituiscono l'header, poi con quelle del corpo del messaggio, e per chiudere il comando di *POST* con una sequenza *CRLF,* un punto, ed ancora un *CRLF.*
Per terminare infine la sessione, è necessario inviare il comando *QUIT,* a cui il server risponderà con il codice 205, che sta ad indicare *Bye Bye.* A grandi linee questo è il protocollo utilizzato per colloquiare con un server di news, naturalmente non abbiamo visto tutti i comandi disponibili nel protocollo standard, né quelli disponibili nelle varie estensioni apportate, che potete eventualmente studiare sugli RFC.

FORMATO DELL'HEADER

L'header di un messaggio è formato da diverse parti, di cui, alcune necessarie ed altre opzionali. Ogni parte è formata da diverse linee in cui abbiamo il nome del campo seguito dai due punti, ed il valore o i valori del campo separati da un punto e virgola. Iniziamo ad esaminare l'*header* a partire dalle parti richieste.
La *Relay Version* identifica il client che spedisce il messaggio sulla rete lungo il link successivo, e deve essere sempre inserito nella prima linea dell'header. Il valore di tale campo è composto da due valori, il primo indica la versione del client, mentre il secondo è il nome di dominio completo del sito.
Ad esempio:

```
Relay Version: version 1.0 beta 01/09/2005 ; site
www.ioprogrammo.net
```

Il campo successivo è la versione del client che ha immesso il messaggio sulla rete, ed il suo formato è identico al campo *Relay Version:*

```
Posting Version: version A; site
www.dotnetarchitects.it
```

La linea *From* identifica il mittente del messaggio , secondo la sintassi *ARPAnet,* che può, oltre all'indirizzo email, includere anche il nome. Le seguenti tre linee sono tutte valide, in diversi formati:

```
From: antonio.pelleriti@ioprogrammo.it
From: antonio.pelleriti@ioprogrammo.it
 (Antonio Pelleriti)
From: Antonio Pelleriti
<antonio.pelleriti@ioprogrammo.it>
```

Il campo *Date* identifica la data di spedizione del messaggio, secondo diverse convenzioni possibili, ma che in genere viene scritta secondo la seguente sintassi:

```
Date: Weekday, DD-Mon-YY HH:MM:SS TIMEZONE
```

In cui *TIMEZONE* è in genere scritta secondo la classica abbreviazione a tre lettere delle zone mondiali, ad esempio GMT, PST, PDT, e così via.
Ad esempio:

```
Date: Monday, 05-Sep-05 12:15:34 GMT
```

A questo punto è necessario indicare i newsgroup a cui si spedisce il messaggio, separati dalla virgola. Se l'elenco contiene newsgroup non esistenti, essi verranno ignorati dai server.

```
Newsgroups: it.test,
microsoft.public.dotnet.it.csharp, it.discussioni.auto
```

Ogni messaggio deve essere identificato da un ID univoco, che viene indicato secondo la seguenti sintassi:

```
Message-ID: <stringasenzaspazi@dominio>
```

Dove il dominio è quello dell'host che ha spedito il messaggio. La stringa prima del carattere @, non può contenere spazi né < o >, e può essere formata da numeri oppure da lettere in genere originati dalla data di spedizione, ma la lunghezza può essere variabile. Infine l'oggetto, o titolo del messaggio, è indicato dal campo *Subject:*

```
Subject: oggetto del messaggio
```

I campi opzionali sono invece *Followup-To, Date-Received, Expires, Reply-To, Sender, References, Control, Distribution, Organization.* In particolare nel nostro newsreader di esempio utilizzeremo il campo *References*, che viene utilizzato quando il messaggio da postare è una risposta ad un messaggio precedente. In questo caso il *Subject* inizia di default con la stringa *"Re: "* seguita dall'oggetto del messaggio a cui si risponde. Il campo *References* riporterà invece il *MessageID* originale, compreso fra < >.
Ad esempio:

```
References: <stringasenzaspazi@dominio>
```

Un campo quasi sempre presente è anche *Sender,* che rappresenta il mittente responsabile della spedizione del messaggio, e quindi può anche essere diverso da quello indicato in *From.* Ad esempio se *tizio@pippo.it* vuol inviare un messaggio, utilizzando l'account *caio@sito.it,* avremo entrambi i campi, ma il server verificherà solo che il campo *Sender* abbia i privilegi per postare:

```
From: tizio@pippo.it
Sender: caio@sito.it
```

PROGRAMMAZIONE DI RETE

Per stabilire una connessione TCP con un qualunque computer collegato ad una rete, di cui conosciamo l'indirizzo o il nome, e il numero della porta aperta ad un dato servizio, il framework .NET mette a disposizione la classe *TcpClient.* Tramite un oggetto *TcpClient,* dopo la connessione, è possibile spedire e ricevere dati sulla connessione in maniera sincrona bloccante, quindi dobbiamo rispettare rigidamente il protocollo di comunicazione, sapendo quando ci sono dati da leggere e quando invece il computer dall'altro lato della connessione si aspetta che siamo noi a spedire qualcosa.
Il seguente esempio mostra come si stabilisce una connessione

```
static void Connect(string server, int port, string
                                        message)
{
   try
   {
      TcpClient client = new TcpClient(server, port);
      // traduce il messaggio in ASCII e lo trasforma
      // in array di byte.
      Byte[] data = System.Text.Encoding.ASCII
                               .GetBytes(message);
      NetworkStream stream =client.GetStream();
      // spedisce il messaggio.
      stream.Write(data, 0, data.Length);
      // Buffer per memorizzare la risposta.
      data = new Byte[256];
      // risposta in formato string ASCII
      string responseData = String.Empty;
      // legge i byte di risposta.
      int bytes = stream.Read(data, 0, data.Length);
      responseData = System.Text.Encoding.ASCII
                        .GetString(data, 0, bytes);
      Console.WriteLine("ricevuto: {0}", responseData);
      client.Close();
   }
   catch (ArgumentNullException e)
   {
     Console.WriteLine("ArgumentNullException:
                                       {0}", e);
   }
   catch (SocketException e)
   {
     Console.WriteLine("SocketException: {0}", e);
   }
}
```

Una procedura simile verrà usata dal nostro client per i gruppi di discussione, ad esclusione del fatto che invece di scrivere e leggere byte per byte, utilizzeremo le classi *StreamReader* e *StreamWriter* per scrivere e leggere direttamente linee intere di testo, cioè sequenze chiuse dai caratteri *"\r\n"*, e ciò si ottiene incapsulando l'oggetto *NetworkStream* ricavato dal metodo *GetStream* della classe *TcpClient,* ad esempio:

```
StreamWriter writer=new
          StreamWriter(tcpClient.GetStream());
writer.WriteLine("LIST");
```

in questo modo spediremo al computer dall'altro lato la sequenza *"LIST\r\n"*, cioè uno dei comandi NNTP che abbiamo già incontrato.

CONNETTERSI AL SERVER

Implementiamo una classe *Session* che rappresenta una sessione di colloquio con un server di

NOTA

NEWS SERVERS

Esistono server di news gratuiti, e che non necessitano di autenticazione, potete provare ad esempio quelli del provider con cui vi connettere ad internet, come news.libero.it, news.tim.it, od ancora il server di microsoft msnews.microsoft.com , oppure, per testare l'applicazione di esempio o per sviluppare la vostra potete utilizzare un server in locale. Per l'articolo è stato utilizzato ad esempio Coffee Link Open News Server, che è liberamente scaricabile da http://sourceforge.net/projects/clnews

news, dotandola dei campi server e port, che rappresentano il nome del server e la porta di connessione, username e password per l'eventuale autenticazione, e di un campo booleano postingAllowed per memorizzare la possibilità o meno di postare messaggi per mezzo del server stesso.La classe *Session* inoltre, utilizzerà un oggetto *TcpClient* per connettersi al server, specificando appunto il nome dell'host e la porta, che di default come detto è la 119.

Il costruttore della classe *TcpClient* che accetta questi due parametri infatti apre direttamente la connessione e, come prima operazione, non dobbiamo far altro che verificare che la risposta contenga il codice 200. Ciò avviene nel metodo *Open*:

```
public void Open()
{
    tcpClient = new TcpClient(server, port);
    NetworkStream stream=tcpClient.GetStream();
    StreamReader reader = new StreamReader(stream);
    string responseData=reader.ReadLine();
    NNTPStatusCode status =
                    GetResponseStatusCode(responseData);
        if (status ==
            NNTPStatusCode.ServerReadyPostingAllowed)
                postingAllowed = true;
        if (status ==
            NNTPStatusCode.ServerReadyNoPostingAllowed)
                postingAllowed = false;
        ...
}
```

Per leggere i dati dalla connessione di rete utilizziamo la classe *NetworkStream*, in combinazione con uno *StreamReader* per leggerli riga per riga.

Dopo aver letto dunque la prima linea di testo restituita dal server, il metodo *GetResponseStatusCode*, ne effettua il parsing, verificando la presenza del codice 200 e se sia consentito o meno il posting di nuovi messaggi.

```
public NNTPStatusCode
                GetResponseStatusCode(string status)
{
    int code =
                Int32.Parse(status.Substring(0,3));
    string text = (status.Length >
                3)?status.Substring(4):"";
    if (Enum.IsDefined(typeof(NNTPStatusCode),
                code))
        return (NNTPStatusCode)code;
    else
        throw new
        ArgumentOutOfRangeException("Invalid return code
                - " + status.Substring(0,3));
}
```

Non tutti i server richiedono l'autenticazione, quindi abbiamo lasciato libera la possibilità di utilizzare username e password, per connettersi al server.

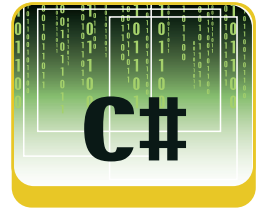
Ancora nel metodo *Open*, dopo lo stabilimento della connessione, e se i campi della classe *Session* username e password sono stati inizializzati, viene invocato il metodo *Autenticate*, che però scopriremo come funziona dopo aver esposto come implementare la gestione dei comandi e delle risposte previsti dal protocollo NNTP.

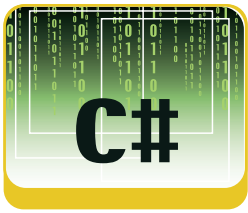
COMANDI E RISPOSTE

Per l'implementazione dei comandi NNTP che serviranno al nostro newsreader per dialogare con il server, faremo ricorso ai classici concetti di programmazione ad oggetti. Ogni comando sarà dunque rappresentato da una classe derivata da una generica *Command* astratta, da cui ogni classe figlia erediterà il generico metodo *Send*. Esso non farà altro che spedire sulla connessione di rete una stringa di comando NNTP, memorizzata nel campo di classe *command*.

```
public void Send()
{
    stream=tcpClient.GetStream();
    writer = new StreamWriter(stream);
    writer.AutoFlush=true;
    string strCmd=command.ToString();
    if (parameters!=null && parameters.Length != 0)
    strCmd += " " + String.Join(paramSeparator,
                        parameters);
    writer.WriteLine(strCmd);
    ReadResponse();
}
```

Dopo aver spedito il comando, bisognerà leggere la risposta del server, cosa che sarà effettuata dal metodo *ReadResponse*, in maniera diversa per ogni comando NNTP. Ad esempio il comando *LIST* si aspetterà una lista di newsgroups, quindi la risposta del server sarà trattata in maniera diversa da quella che ci si aspetta per il comando *GROUP*.

La classe *Command* dichiara dunque un metodo astratto *ReadResponse*, implementato da ogni classe figlia. Ogni comando, come visto è formato da una o più parole, che sarà memorizzata nel campo *string command*. Inoltre ogni comando potrà avere o meno dei parametri; quindi, ancora nella classe *Command* avremo un campo parameters, di tipo *string[]*, che verrà riempito eventualmente con i parametri del comando stesso. La risposta ad un comando NNTP sarà rappresentata da una classe *Response*, caratteriz-

LE RFC

Il protocollo NNTP originale è specificato nell'RFC 977. Poiché tale documento è stato creato quasi 20 anni fa, e da allora il protocollo NNTP, con la diffusione dei newsgroup, è diventato sempre più utilizzato, si sono rese necessarie varie aggiunte e modifiche alla specifica stessa. Ciò ha portato alla scrittura ed implementazione di estensioni più o meno ufficiali, da parte degli sviluppatori di newsreader, o di news server, o di chiunque si trovasse a lavorare con il protocollo. Tali estensioni, anche senza portare alla stesura di una nuova specifica, sono ormai di uso comune, e quindi sono stati riassunte nell'RFC 2980, dal titolo Common NNTP Extensions.

zata dal contenuto completo della risposta, e dal codice di stato.

LA CLASSE LISTCOMMAND

Vediamo fra le tante, la classe *ListCommand*, che rappresenta appunto il comando *GROUP*. Ogni classe derivata da *Command*, avrà sempre lo stesso funzionamento, in particolare basterà implementare il metodo *ReadResponse*, in maniera da leggere adeguatamente la risposta del server, mentre nel costruttore verrà specificato il comando testuale da inviare e gli eventuali parametri. In questo caso non abbiamo parametri, quindi non sarà inizializzato il campo *parameters*, ed il costruttore sarà fatto così:

```
public ListCommand(Session sess):base(sess)
{
        this.command=NNTPCommand.List;
        groupsRetrieved=new GroupCollection();
}
```

La classe *NNTPCommand* è un raccoglitore dei comandi NNTP, che contiene semplicemente delle costanti stringa. Nel nostro esempio i comandi implementati sono:

```
public class NNTPCommand
{
public const string List="LIST";
    public const string Group="GROUP";
    public const string XOver="XOVER";
    public const string Body = "BODY";
    public const string Post = "POST";
    public const string EndPost = ".";
    public const string AuthInfoUser = "AUTHINFO
                                        USER";
     public const string AuthInfoPass = "AUTHINFO
                                        PASS";
        public const string Quit = "QUIT";
}
```

Per chi volesse implementare altri o addirittura tutti i comandi NNTP, basterà aggiungerli fra queste costanti e fornire la relativa implementazione figlia della classe *Command*. Ogni classe necessita naturalmente di diverse informazioni o metodi per svolgere il proprio compito. La classe *ListCommand* ad esempio, dovrà memorizzare l'elenco di gruppi restituito dal server, e quindi è stato previsto un campo *groupsRetrieved* di classe *GroupCollection*, anch'esso inizializzato nel costruttore.

Tralasciamo l'implementazione della classe *GroupCollection*, che esula dai nostri scopi, rimandando al codice in allegato all'articolo. Inoltre, dato che il processo di lettura della risposta, in questo caso potrebbe essere lungo, sarà meglio utilizzare un diverso thread per non bloccare l'applicazione, ed invece di bloccare l'applicazione fino a che l'intero elenco non sia stato scaricato, stampare i singoli newsgroup uno per uno, mano a mano che verranno letti.

Ciò suggerisce l'idea di utilizzare un evento, che sarà scatenato appunto alla ricezione di ogni gruppo:

```
public event NewsgroupRetrieveHandler
                                NewsgroupRetrieved;
```

Il delegate *NewsgroupRetrieveHandler* è dichiarato in questa maniera:

```
public delegate void
    NewsgroupRetrieveHandler(NewsgroupEvent ev);
```

e sarà utilizzato nella nostra applicazione Windows per ricavare informazioni sul newsgroup scaricato, informazioni contenute in un oggetto di classe *NewsgroupEvent*:

```
public class NewsgroupEvent:EventArgs
{
    private NewsGroup group;
    public NewsgroupEvent(NewsGroup gr):base()
    {
        this.group=gr;
    }
    public NewsGroup Group
    {
        get{return group;}
    }
}
```

Un oggetto di tale tipo sarà creato nel metodo *ReadResponse* della classe *ListCommand*, per ogni newsgroup scaricato dal server.

Vediamo quindi come nel caso del comando *LIST*, il metodo *ReadResponse* interpreta la risposta del server, e ricava le informazioni sui gruppi.

```
protected override void ReadResponse()
{
  StringBuilder sb=new StringBuilder();
  string name;
  string s = reader.ReadLine();
  // Server response = "215 Newsgroups
  // follow"
  serverResponse=new
  Response(session.GetResponseStatusCode(s));
     NewsGroup group;
     if(serverResponse.StatusCode==
          NNTPStatusCode.ListOfNewsGroupsFollows)
```

BIBLIOGRAFIA

• "C# AND .NET PLATFORM", TROELSEN (APRESS)

• "Applied Microsoft .NET programming", Richter (Microsoft Press)

• "Essential .NET, Volume I : The Common Language Runtime", Don Box , Chris Sells (Addison Wesley)

• RFC 977 NNTP Protocol - http://www.karlsruhe.org/rfc/rfc977.txt

• RFC 2980 Common NNTP Extensions - http://www.karlsruhe.org/rfc/rfc2980.txt

```
    {
        name=s;
        while (name != ".") // "." response
                means that the list comes to the end
        {
            name = reader.ReadLine();
            if (name != ".")
            {
                name = name.Substring(
                                    0,name.IndexOf(' '));
                sb.Append(name+"\r\n");
                group=new NewsGroup(name.Trim());
                groupsRetrieved.Add(
                                    name.Trim(),group);
                this.NewsgroupRetrieved(new
                                NewsgroupEvent(group));
            }
        }
    }
    else sb.Append(s);
    this.serverResponse.Content=sb.ToString();
}
```

Come primo passo si verifica che la prima linea sia una risposta di stato 215, ad indicare che verrà di seguito trasmessa una lista di newsgroup.
Se ciò è vero, vengono lette tutte le linee fino a trovarne una che contiene solo un punto, che indica la fine della lista. Per ogni linea letta, viene semplicemente ricavato il nome del gruppo, creato un oggetto *Newsgroup*, ed una volta aggiunto tale oggetto alla collection *groupsRetrieved* viene anche scatenato l'evento *NewsgroupRetrieved*. Infine l'intera risposta testuale viene conservata nel campo *serverResponse*.

SPEDIRE E RICEVERE MESSAGGI

Per rappresentare un messaggio o articolo di un Newsgroup, implementiamo una classe *Article*, e dato che un articolo è composto da un'intestazione e dal corpo, essa incapsulerà un *ArticleHeader* ed un *ArticleBody*. Facendo riferimento al paragrafo in cui abbiamo mostrato il formato dell'header, implementeremo così una struttura *ArticleHeader:*

```
public struct ArticleHeader
{
        public int LineCount;
        public int ByteCount;
        public string References;
        public string MessageId;
        public string DateString;
        public string From;
        public string Subject;
        public string Newsgroups;
        public override string ToString()
        {
            string relayversion="version 1.0
        beta 01/09/2005 ; site www.ioprogrammo.net";
            string postingversion="version A;
                        site www.dotnetarchitects.it";
            StringBuilder sb=new
            StringBuilder();
            sb.AppendFormat("Relay Version:
            {0}{1}",relayversion,Environment.NewLine);
            sb.AppendFormat("Posting Version:
        {0}{1}",postingversion,Environment.NewLine);
            sb.AppendFormat("From:
                    {0}{1}",From,Environment.NewLine);
            sb.AppendFormat("Date:
            {0}{1}",DateString,Environment.NewLine);
            sb.AppendFormat("Newsgroups:
            {0}{1}",Newsgroups,Environment.NewLine);
            sb.AppendFormat("Subject:
                {0}{1}",Subject ,Environment.NewLine);
            sb.AppendFormat("Message-
        ID: {0}{1}",MessageId,Environment.NewLine);
            if(References!=String.Empty &&
                                    References!=null)
            sb.AppendFormat("References:
            {0}{1}",References,Environment.NewLine);
            return sb.ToString();
        }
}
```

Mentre per il body sarà sufficiente prevedere un campo *string* che contenga il corpo stesso del messaggio. Sarebbe stato sufficiente utilizzare un campo *string* direttamente nella classe *Article*, ma in previsione di futuri miglioramenti, ad esempio per trattare messaggi con il corpo in formati diversi, come l'HTML, si è scelto di incapsulare il campo nella classe *ArticleBody*.

CONCLUSIONI

In questo articolo abbiamo visto come realizzare un *NewsReader* partendo dallo studio del protocollo NNTP ed utilizzando le classi messe a disposizione dal .NET Framework. Naturalmente l'applicazione realizzata non sarà esente da difetti, in quanto creata per esercizio e non destinata al pubblico. Ad esempio non esistono meccanismi di cache dei messaggi scaricati, o dei newsgroup, quindi vengono scaricati dal server ogni volta tutti quanti.
Ma magari, chi avesse il tempo e la passione, potrebbe utilizzarla come punto di partenza per creare qualcosa di più professionale.

Antonio Pelleriti

Potete contattare l'autore per suggerimenti, critiche o chiarimenti all'indirizzo e-mail antonio.pelleriti@ioprogrammo.it, oppure sul sito www.dotnetarchitects.it

**Per l'articolo è stato utilizzato un server di news, gratuito, scritto in java, che potete scaricare gratuitamente, ed utilizzare per le vostre prove avviandolo in locale.
Si tratta di *Coffee Link News Server*, e lo trovate su http://sourceforge.net/projects/clnews.**

SNMP ti dà il controllo totale

Ecco il protocollo che consente di analizzare il comportamento di qualunque periferica hardware. Collegata a un computer, ad una rete o isolata, non fa differenza, noi la controlliamo!

Nel numero 95 di ioProgrammo avevamo iniziato a parlare di SNMP, indicandolo come il protocollo in grado di analizzare informazioni provenienti da qualsiasi periferica Hardware. Ad esempio avevamo indicato SNMP come standard da utilizzare per tenere sotto controllo il traffico passante attraverso la scheda di rete, oppure per conoscere l'occupazione della memoria in tempo reale o quella del disco rigido. La nostra trattazione del tutto teorica spiegava come ciascuna periferica sia dotata di un MIB, un numero univoco che la identifica, e come questo possa essere utilizzato per interrogare la periferica. Avevamo inoltre introdotto una tecnica per mappare eventi gestiti da SNMP in classi interrogabili da WMI la *Windows Management* instrumentation tramite l'utility *smi2smr*. In questo articolo realizzeremo un progetto pratico che utilizza le informazioni precedentemente introdotte, per tenere sotto controllo alcuni comportamenti del sistema operativo.

IL PROGETTO IN VB

Il progetto implementato è costituito da una sola form e da alcuni moduli che raccolgono alcune funzioni utili al programma stesso. In particolare:

- **DISKS:** raccoglie le procedure e le funzioni utili al controllo dello spazio disco libero su tutte le partizioni del proprio disco;
- **PING:** implementa una semplice funzione che consente di "pingare" un host remoto;
- **PORTE:** implementa una procedura che, sfruttando l'*SNMP Provider,* consente il controllo delle porte TCP/IP ed UDP aperte.

Il programma realizzato implementa alcune procedure che, attraverso l'utilizzo di WMI e, in particolare, dell'*SNMP Provider,* consentono di monitorare alcune risorse del computer locale sul quale il programma gira. Fatta eccezione della procedura per il controllo delle interfacce, restano da vedere quelle relative a:

- Controllo occupazione dischi;
- Controllo servizi *down;*
- Controllo processi (creazione ed eliminazione);
- Controllo segnalazioni eventi nell'*Event Viewer.*

All'avvio, l'evento *Load* della form principale compie una serie d'operazioni. Le più importanti riguardano il controllo sulla prima esecuzione del programma e l'impostazione delle variabili che rispecchiano le scelte precedentemente fatte dall'utente.
Se il programma è stato avviato per la prima volta, vengono richiamate una serie di procedure che si preoccupano di creare, all'interno del *Registry,* una struttura che memorizza diverse impostazioni utili al corretto funzionamento dello stesso. La chiave che fa da root a queste informazioni è denominata *Master-*

COME FUNZIONA WMI?

WMI è l'implementazione Microsoft di WBEM. WBEM è un'iniziativa industriale per lo sviluppo di una tecnologia standard di accesso alle informazioni. WMI funziona per mezzo di *"provider"*, ovvero DLL contenute nella cartella *%SystemRoot%\System32\WBEM* che fanno da intermediari tra il sistema e le applicazioni. Ad esempio quando il CIM riceve una richiesta non direttamente gestibile da WMI questa viene passata al provider interessato.

Agent (sotto *HKEY_CURRENT_USER\Software\VB and VBA Program Settings)* ed al suo interno troviamo le seguenti sottochiavi:

- **Drives:** per il controllo dello spazio disco sulle partizioni;
- **Eventi:** per il controllo di nuovi eventi nell'*Event Viewer;*
- **Install:** per verificare se si tratta di primo avvio del programma;
- **Interfacce:** per il controllo di interfacce di rete;
- **Processi:** per il controllo sulla creazione o eliminazione di processi;
- **Servizi:** per il controllo di servizi che vanno giù.

Fatta eccezione per la sottochiave *Drives* ed *Install,* le altre controllano semplicemente se l'utente ha o meno attivato quel determinato tipo di controllo e lo inseriscono all'avvio. La chiave *Drives,* invece, conserva anche altre informazioni tra le quali: la soglia di allarme sullo spazio disco rimasto libero, espresso come valore percentuale. Una volta determinate le impostazioni lette all'interno del *Registry,* attraverso un'apposita procedura denominata *ReadFromREG(),* l'evento *Load* della form principale inizia ad impostare le variabili ed i controlli all'interno della form. Successivamente, valorizza un'array denominato *WQLQuery* (che non fa altro che memorizzare le varie stringhe WQL che ci serviranno per i controlli), imposta i flag relativi all'attivazione delle procedure (lanciandole se necessario) e visualizza, finalmente, la form.
Essa può essere suddivisa in almeno tre parti principali:

- Una lista di pulsanti alla sinistra;
- Una serie di pannelli sovrapposti nella parte centrale;
- Una serie di led nella parte inferiore sinistra.

Ogni pulsante attiva un determinato pannello (attraverso il metodo *ZOrder*) che consente di visualizzare eventuali allarmi scattati, impostare o meno il controllo ed eliminare i log creati. Passiamo quindi al primo di questi controlli ossia il controllo sui servizi.

IL CONTROLLO DEI SERVIZI

Il controllo dei servizi è implementato attraverso l'utilizzo di due procedure: *ServiceStatus()* e *ServiceSink_OnObjectReady()*. Dall'analisi di queste due procedure, avremo modo di comprendere la maggior parte delle restanti poiché, nella maggior parte dei casi, utilizzano la stessa tecnica.

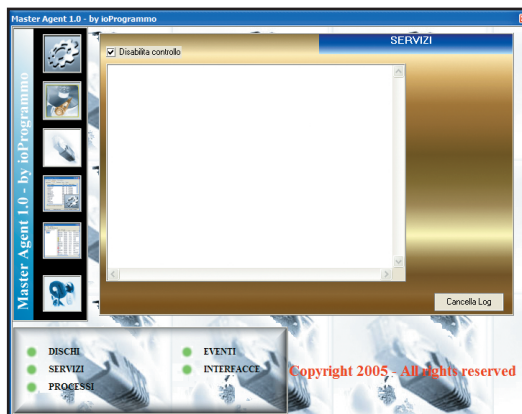



Fig. 1: *L'aspetto finale della nostra applicazione*

Eccole di seguito:

```
Public Sub ServiceStatus()

  Dim objWmiLocator As New
                        WbemScripting.SWbemLocator
  Dim objWmiServices As SWbemServices

  'Connessione al namespace della macchina locale
    Set objWmiServices=
      objWmiLocator.ConnectServer(".", "root\CIMV2")
    objWmiServices.Security_.ImpersonationLevel=3
    objWmiServices.Security_.Privileges.AddAsString
                                ("SeSecurityPrivilege")
    Set ServiceSink=New SwbemSink

  'Controlla eventuali servizi down
    'WQLQuery="SELECT * FROM
        __InstanceModificationEvent WITHIN 1 WHERE
              TargetInstance ISA 'Win32_Service'AND
    (PreviousInstance.State <> TargetInstance.State)
                AND TargetInstance.State='Stopped'"
    objWmiServices.ExecNotificationQueryAsync
                            ServiceSink, WQLQuery(5)
```

COME SI INSTALLA WMI?

I provider WMI vengono installati insieme al Sistema Operativo, ma la lista di quelli messi a disposizione dal sistema può essere allungata attraverso la definizione di nuovi provider proprietari. Abbiamo già detto che il provider SNMP consente di fare da ponte tra WMI e le periferiche da controllare.

È anche vero che WMI deve disporre di una serie di classi in grado di rappresentare i MIB delle periferiche da controllare. Per inserire un database di MIB all'interno del CIM comprensibile da WMI è necessario utilizzare l'utility smi2smir nella forma smi2smir /a <nome file mib>

```
End Sub

Public Sub ServiceSink_OnObjectReady(ByVal
    StatusEvent As SWbemObject, ByVal EventContext
                          As SWbemNamedValueSet)

    Dim ListItem
    Dim i As Integer

    ListItem=Split(StatusEvent.GetObjectText_, Chr(10))

    'Crea la stringa per il log
      txtServices.Text=txtServices.Text +
        Trim(Str(Date)) + " " + Trim(Str(Time)) + " " +
      Mid(ListItem(13), 16, Len(ListItem(13)) - 16) + "
                                        down" + vbCrLf

    'Attiva led ROSSO
      LedServicesRED.ZOrder 0

End Sub
```

Dall'analisi della prima procedura, si evincono subito alcune particolarità. La prima è l'utilizzo delle istruzioni:

```
objWmiServices.Security_.ImpersonationLevel=3
objWmiServices.Security_.Privileges.AddAsString
                                ("SeSecurityPrivilege")
```

che consentono di definire le modalità di accesso, per motivi di sicurezza, alle informazioni che stiamo cercando di ottenere. Non entreremo nel merito di questo argomento e rimandiamo il tutto alla documentazione Microsoft, abbastanza esauriente.
L'unica cosa da sottolineare è il fatto che, in molti casi istruzioni come quelle appena mostrate, sono assolutamente necessarie per impedire errori del tipo *"Accesso negato"*.
La seconda particolarità è il modo con cui viene avviata la query *WQL*. Analizziamola:

```
"SELECT * FROM __InstanceModificationEvent
                    WITHIN 1 WHERE TargetInstance ISA
       'Win32_Service'AND (PreviousInstance.State <>
                               TargetInstance.State) AND
                         TargetInstance.State='Stopped'"
```

Innanzitutto notiamo subito l'utilizzo di *__InstanceModificationEvent*. La classe appena menzionata consente di tener traccia di eventuali modifiche avvenute all'interno di altre. Ad esempio, nel caso specifico, stiamo "chiedendo" di essere avvertiti quando avviene un cambiamento all'interno di dell'istanza *Win32_Service*. La clausola *WITHIN* costituisce l'intervallo di *polling*, mentre *TargetInstance* permette di specificare l'istanza della classe da controllare.

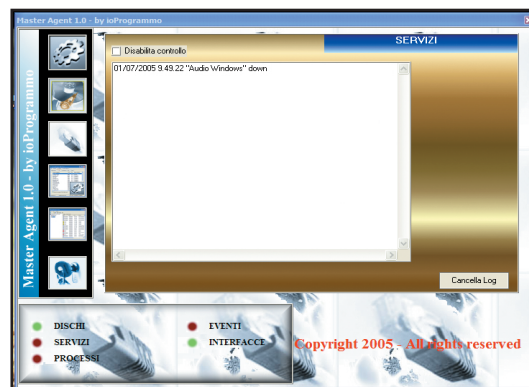



Fig. 2: Il pannello per il controllo dei servizi

Se ci fermassimo alla stringa prima dell'*AND*, la query controllerebbe "qualunque" cambiamento nell'istanza *Win32_Service*, impegnando notevolmente le risorse di sistema. Tuttavia, per restringere il campo d'azione e, quindi, sacrificare meno risorse, possiamo migliorare la query attraverso la seconda parte della stringa ossia:

```
(PreviousInstance.State <> TargetInstance.State)
                    AND TargetInstance.State='Stopped'"
```

che, in parole semplici, chiede di monitorare ogni cambiamento avvenuto all'interno di *Win32_Service*, ma considerando solo i casi in cui ci sia un cambio di stato di un qualunque servizio e lo stato attuale sia pari a *Stopped*.
Le due proprietà *PreviousInstance* e *TargetInstance* della *__InstanceModificationEvent* rappresentano lo stato precedente ed attuale dell'istanza della classe e possiamo considerarle quasi indispensabili per gli scopi che ci siamo prefissi.
A questo punto, però, non abbiamo risolto del tutto il nostro problema ossia quello di essere avvertiti quando la query impostata è soddisfatta. A questo proposito entra in gioco un nuovo oggetto: l'*SWBemSink* attraverso il quale è possibile definire le procedure che "riceveranno" tali notifiche.
Attraverso *WMI* sono riconosciute, per questi scopi, quattro procedure predefinite: *OnCompleted, OnObjectPut, OnObjectReady* e *OnProgress.*
Di queste verrà gestita solo la *OnObjectReady* che viene chiamata in causa non appena l'evento, definito dalla query *WQL,* è "pronto".
Ritornando quindi alle due procedure di cui sopra, dovrebbe essere ora più chiaro il codice implementato.In particolare, attraverso:

```
objWmiServices.ExecNotificationQueryAsync
```

```
ServiceSink, WQLQuery(5)
```

lanciamo la nostra query asincrona (sfruttando l'appropriato metodo *ExecNotificationQueryAsync* di *objWmiServices*) specificando, però, come primo parametro, un oggetto *SWBemSink* attraverso i cui eventi gestiremo gli allarmi.
Nel momento in cui la query è soddisfatta (ossia, nel nostro caso, un servizio viene fermato), "scatta" l'evento *ServiceSink_OnObjectReady()* che ci consentirà di gestire l'allarme.
In particolare, il codice implementato agisce sul primo parametro passato alla procedura ossia *StatusEvent* che, nel caso specifico della classe *Win32_Service*, in corrispondenza dello stop del servizio *Centro Sicurezza PC*, ha questa struttura:

```
instance of __InstanceModificationEvent
{
   PreviousInstance=instance of Win32_Service
{
   AcceptPause=FALSE;
   AcceptStop=TRUE;
   Caption="Centro sicurezza PC";
   CheckPoint=0;
   CreationClassName="Win32_Service";
   Description="Effettua il monitoraggio delle
        impostazioni e delle configurazioni di protezione
                                        del computer.";
   DesktopInteract=FALSE;
   DisplayName="Centro sicurezza PC";
   ErrorControl="Normal";
   ExitCode=0;
   Name="wscsvc";
   PathName="C:\\WINDOWS\\System32
                        \\svchost.exe -k netsvcs";
   ProcessId=968;
   ServiceSpecificExitCode=0;
   ServiceType="Share Process";
   Started=TRUE;
   StartMode="Auto";
   StartName="LocalSystem";
   State="Running";
   Status="OK";
   SystemCreationClassName=
                        "Win32_ComputerSystem";
   SystemName="PRESARIO";
   TagId=0;
   WaitHint=0;
};
TargetInstance= instance of Win32_Service
{
   AcceptPause=FALSE;
   AcceptStop=FALSE;
   Caption="Centro sicurezza PC";
   CheckPoint=0;
   CreationClassName="Win32_Service";
   Description="Effettua il monitoraggio delle
        impostazioni e delle configurazioni di protezione
                                        del computer.";
   DesktopInteract=FALSE;
   DisplayName="Centro sicurezza PC";
   ErrorControl="Normal";
   ExitCode=0;
   Name="wscsvc";
   PathName="C:\\WINDOWS\\System32
                        \\svchost.exe -k netsvcs";
   ProcessId=0;
   ServiceSpecificExitCode=0;
   ServiceType="Share Process";
   Started=FALSE;
   StartMode="Auto";
   StartName="LocalSystem";
   State="Stopped";
   Status="OK";
   SystemCreationClassName=
                        "Win32_ComputerSystem";
   SystemName="PRESARIO";
   TagId=0;
   WaitHint=0;
};
TIME_CREATED="127646270468013504";
};
```

Attraverso l'utilizzo della funzione *Split()*, quindi, non facciamo altro che spezzare nelle varie righe l'intero testo e prelevare solo le parti che ci interessano.

IL CONTROLLO DEI PROCESSI E DEGLI EVENTI

Malgrado la tecnica utilizzata per controllare i processi e gli eventi sia simile alla precedente, occorre precisare qualche dettaglio che la contraddistingue dalla precedente.

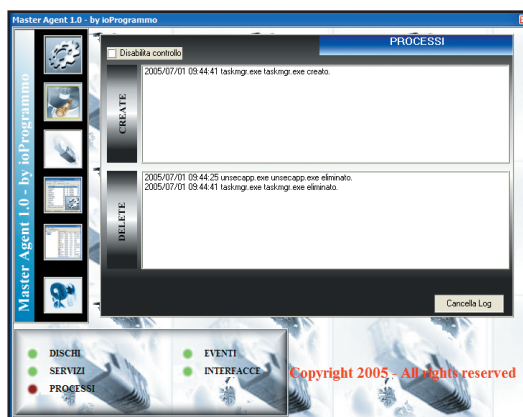



Fig. 3: ***Il pannello di controllo dei processi***

Innanzitutto va detto che le procedure implementate consentono di controllare sia la creazione che la "distruzione" di qualunque processo. La procedura principale è la seguente:

```
Public Sub ControlloProcessi()

    Dim objWmiLocator As New
                            WbemScripting.SWbemLocator
    Dim objWmiServices As SWbemServices

    Set ProcessCreateSink=New SWbemSink
    Set objWmiServices=
        objWmiLocator.ConnectServer(".","root\CIMV2")
    objWmiServices.Security_.ImpersonationLevel=3

    'WQLQuery="SELECT * FROM
              __InstanceCreationEvent WITHIN 1 WHERE
                   TargetInstance ISA 'Win32_Process'"
     objWmiServices.ExecNotificationQueryAsync
                         ProcessCreateSink, WQLQuery(2)

     Set ProcessDeleteSink=New SWbemSink
     Set objWmiServices=
        objWmiLocator.ConnectServer(".","root\CIMV2")
     objWmiServices.Security_.ImpersonationLevel=3

     objWmiServices.ExecNotificationQueryAsync
                          ProcessDeleteSink,WQLQuery(3)

End Sub
```

La principale differenza con la precedente è l'utilizzo di due oggetti *SWBemSink*, *ProcessCreateSink* e *ProcessDeleteSink* e, di conseguenza, di due query *WQL*. L'estrazione delle informazioni è analoga alla procedura precedente. Il controllo degli eventi è anch'esso simile a quanto appena visto, fatta eccezione per il fatto che vengono separati i vari eventi a seconda se si tratti di *Errori, Informazioni* o *Warning*.

IL CONTROLLO DEI DISCHI

Siamo quasi alla fine di questo percorso. Quello che ci resta ora da vedere è la gestione degli allarmi sullo spazio disco delle partizioni del sistema controllato. Cominciamo dunque con il controllo dei dischi.
All'avvio del programma, vengono inizializzate alcune variabili che servono, tra le altre cose, ad impostare correttamente i parametri di allarme sui dischi. Il frammento della procedura *Load* della form principale che si occupa di far questo è il seguente:

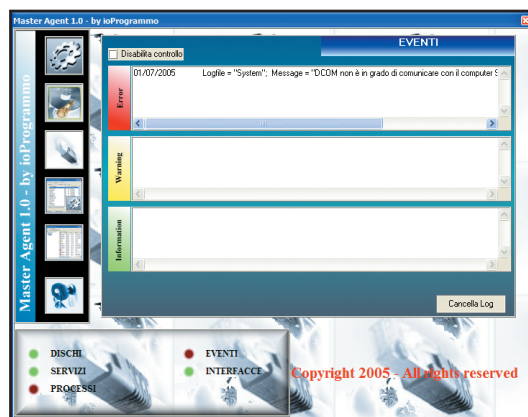



Fig. 4: **Il pannello di controllo degli eventi**

```
' Imposta a false il flag che identifica una
  ' modifica sulla soglia di una partizione
    ThresholdDiskFlag = False

  ' Carica le impostazioni dei drive
    ReadFromREG ("DRIVES")
    CaricaDrives
  ' Imposta i dati sulla prima partizione
    DrvNDX = 0
    frmPrincipale.cmbDisks.ListIndex = 0
    frmPrincipale.SogliaDisk.Value = Drives(0, 1)
    frmPrincipale.lblSogliaDisco =
                     frmPrincipale.SogliaDisk.Value & "%"
```

La variabile *ThresholdDiskFlag* consente di tener traccia di eventuali cambiamenti alle soglie di allarme dei dischi permettendo di richiedere all'utente il salvataggio all'interno del registry. Le procedure *ReadFromREG()* e *CaricaDrives()*, consentono di leggere le impostazioni dal registry e preparare l'interfaccia grafica con tutti i parametri corretti (numero di partizioni, soglie di allarme, ecc.).
In particolare, la procedura *CaricaDrives()* è così costituita:

```
Public Sub CaricaDrives(Optional ScriviReg As Boolean)
' Si occupa di cercare tutti i drive(partizioni) presenti
' e valorizzare la combobox cmbDisks con le lettere di
` unità trovate
  Dim objWmiLocator As New
                          WbemScripting.SWbemLocator
  Dim objWmiServices As SWbemServices
  ' Connessione al namespace della macchina locale
    Set objWmiServices =
      objWmiLocator.ConnectServer(".", "root\CIMV2")
    objWmiServices.Security_.ImpersonationLevel = 3
  ' Ricava tutte le partizioni disponibili
    Set Diskset = objWmiServices.ExecQuery(
               "SELECT Name, Size, FreeSpace FROM
          Win32_LogicalDisk WHERE DriveType=3")
  ' Carica la lista delle lettere di unità trovate
    NumDrives = 0
```

```
    For Each disk In Diskset
        frmPrincipale.cmbDisks.AddItem disk.Name
        NumDrives = NumDrives + 1
    Next
  frmPrincipale.cmbDisks.ListIndex = 0
End Sub
```

Attraverso essa, possiamo ricavare tutte le lettere di unità dei soli dischi rigidi (vedi *DriverType=3* nella query) oltre alla dimensione ed allo spazio disco di ognuna di esse.

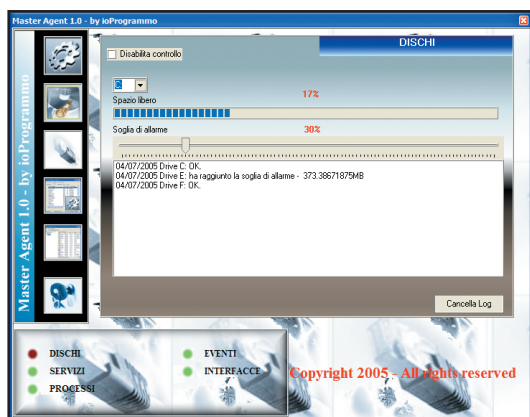



Fig. 5: ***Il pannello di controllo dello spazio disco***

Ovviamente, potevamo ottenere questi dati in altra maniera, ma si è pensato di sfruttare WMI per sottolineare ancora una volta la sua potenza.
Una volta ottenute queste informazioni, se le impostazioni dell'utente prevedono il controllo dello spazio disco, viene lanciata la procedura *ControlloSpazioDisco()*, ad intervalli prestabiliti. Questa procedura è molto simile alle precedenti, fatta eccezione per il fatto che non si tratta di query asincrone.

```
Public Sub ControlloSpazioDisco()
' Si occupa di calcolare la percentuale di spazio disco
` disponibile confrontandola con la soglia di allarme
` fissata.
  Dim objWmiLocator As New
                    WbemScripting.SWbemLocator
  Dim objWmiServices As SWbemServices
  Dim DiskNdx As Integer
  DiskNdx = 0
  ' Connessione al namespace della macchina locale
    Set objWmiServices =
      objWmiLocator.ConnectServer(".", "root\CIMV2")
    objWmiServices.Security_.ImpersonationLevel = 3
  ' Ricerca tutte le partizioni disponibili
    Set Diskset = objWmiServices.ExecQuery(
                "SELECT Name, Size, FreeSpace FROM
           Win32_LogicalDisk WHERE DriveType=3")
  ' Calcola lo spazio disco disponibile e valuta la
  ` soglia d'allarme
    For Each disk In Diskset
      ' Aggiorna l'array dello spazio disco inutilizzato
        DrivesFreeSpace(DiskNdx) = (disk.freespace
                              * 100) / disk.Size
      If Int((disk.freespace * 100) / disk.Size)
                    < Int(Drives(DiskNdx, 1)) Then
          frmPrincipale.lstDisks.AddItem Str(Date) + "
            Drive " + disk.Name + " ha raggiunto la
          soglia di allarme - " + Str((disk.freespace
                           / 1024) / 1024) + "MB"
          ' Attiva led ROSSO
            frmPrincipale.LedDisksRED.ZOrder 0
      Else
          frmPrincipale.lstDisks.AddItem Str(Date) + "
                       Drive " + disk.Name + " OK."
      End If
      DiskNdx = DiskNdx + 1
    Next
    ' Aggiorna la barra che mostra tale spazio
      frmPrincipale.ProgressBar1.Value =
                           DrivesFreeSpace(DrvNDX)
      frmPrincipale.lblFreeSpaceDisk.Caption =
               Str(DrivesFreeSpace(DrvNDX)) + "%"
End Sub
```

Dopo quanto detto, non dovrebbe essere complicato comprendere il listato precedente. L'unica cosa da sottolineare è l'utilizzo dell'array *Drives* che, per ogni partizione trovata, memorizza di volta in volta, la soglia di allarme prescelta.

CONCLUSIONI

Finalmente, siamo arrivati alla fine di questo lungo cammino che, speriamo, sia stato interessante. Ancora una volta, però, desideriamo porre l'accento su un aspetto dell'articolo piuttosto ovvio: la complessità. Purtroppo i concetti, gli esempi e quant'altro ruoti attorno a WMI (così come SNMP) sono tantissimi e non sarebbero bastati intere stesure per scriverli. Consigliamo quindi di avvalersi dei link suggeriti e di eventuale altra documentazione per completare, almeno in parte, questo argomento. Prima di lasciarvi, ricordo che allegate all'articolo ci sono altre due procedure non citate nel progetto, ma che potrebbero essere molto utili.
La prima, *PingHost()*, contenuta nel modulo *PING.bas*, consente di effettuare il ping, tramite WMI, ad un host.
La seconda, *PortsCheck()*, contenuta all'interno del modulo *PORTE.bas*, consente di controllare le porte TCP/IP e UDP aperte sul PC controllato.

Francesco Lippo

Bonjour, la rete è fatta!

Alla scoperta di "Bonjour" una libreria prodotta da Apple che ci consente di creare reti "Autoconfiguranti". In questo appuntamento creeremo un servizio di File Sharing molto particolare…

Tutto inizia da *Zeroconf* o *Zero Configuration Networking*. Questo protocollo, dal nome così evocativo è uno standard dell'IETF, che consente di collegare un computer ad una rete mediante cavo ethernet e vederlo funzionare senza nessun altro intervento, in pratica il sogno di ogni utente medio. *Zeroconf* fa sì che, non dobbiate configurare manualmente DHCP, indirizzo IP o DNS, l'atto di collegare il cavo alla rete, dovrebbe essere sufficiente a due computer per comunicare. Questa meraviglia di invenzione è stata tirata fuori dal cilindro di Apple nel tentativo di favorire il passaggio dalle reti AppleTalk a quelle basate su IP. L'implementazione di questa tecnica è stata denominata: *"RendezVous"*. Successivamente, a causa di alcune beghe legali dovute all'esistenza di un marchio analogo registrato da *Tibco* la tecnologia è stata rinominata *"Apple Bonjour"*. Attualmente *bonjour /zeroconf* è disponibile, come ovvio, per piattaforma Macintosh ma può facilmente essere implementata anche su Windows.
Nel dettaglio, le funzioni offerte da *Bonjour* sono le seguenti:

- Allocazione di IP dinamici senza bisogno di un DHCP server.
- Traduzione di IP in nomi senza bisogno di un DNS server.
- Discovery dei servizi senza bisogno di una directory che li esponga.
- Allocazione di IP Multicast senza bisogno di un MADCAP server.

In questo articolo sfrutteremo il *"Discovery dei servizi"* per creare un servizio di *"File Sharing"* in una nostra rete locale. L'idea è semplice: in ciascun pc collegato alla rete girerà il nostro applicativo. Il software in questione svolgerà i seguenti compiti:

COME INIZIARE

Non c'è necessità di installazione, l'SDK con librerie C e Java è disponibile all'indirizzo http://developer.apple.com/networking/bonjour/

Ovviamente per provare la tecnica dovete avere installato un J2SE. Come ambiente di sviluppo abbiamo utilizzato Eclipse.

- Esporre un manifesto dei servizi offerti
- Effettuare lo scanner della rete per trovare altri host che espongono servizi
- Indicizzare documenti mp3 o altri tipi di file
- Consentire agli host interessati di scaricare dal pc gli mp3 indicizzati
- Scaricare documenti da altri host che espongono lo stesso servizio.

In tutto questo *Bonjour* ci verrà in aiuto, quando si tratterà di trovare gli host della rete che espongono il manifesto dei servizi. Per quanto riguarda invece l'implementazione dell'architettura client/server, che consente l'indicizzazione e il download dei documenti, chiaramente dovremo implementare il protocollo *from scratch*. Di fatto *Bonjour* ci offrirà solo la funzionalità di discovery e non quella di file sharing. La differenza fondamentale, rispetto a qualunque altro tipo di implementazione è che non avremo bisogno di una directory centrale, che tenga aggiornata la lista degli host che espongono il servizio, viceversa sarà bonjour ad andare a caccia nella rete dei vari PC consultabili.

CREAZIONE DEL MANIFESTO

Il primo passo da compiere è "Creare il manifesto". All'interno di questo manifesto, do-

vranno comparire alcuni parametri, come ad esempio il nome del servizio esposto, la porta su cui gira, altre informazioni aggiuntive che analizzeremo nel dettaglio da qui a breve.
La classe fondamentale, attraverso la quale passano tutte le operazioni che comunicano qualcosa con il sistema è DNSSD. In questa classe, vengono definiti tutti i metodi che utilizzeremo. In generale ciascuno di questi metodi, prende in input oltre ad alcuni parametri che servono per l'implementazione delle funzioni, anche un'interfaccia.
Nell'interfaccia in questione dovranno essere definiti dei metodi che restituiscano l'esito di una determinata operazione.
Tutto sarà più chiaro, guardando il seguente esempio:

```
DNSSDRegistration registration;
Registrazione registrazione;
registrazione=new Registrazione();
registration=DNSSD.register("iop2p","_iop2p._tcp",3
                                    0330,registrazione);
```

Lo spezzone di codice di cui sopra, espone il “manifesto” del servizio. Il nome del servizio è *“iop2p”*, il nome del protocollo che utilizzeremo sarà *“_iop2p_ tcp”* e in questo includiamo anche che tale protocollo si affiderà come layer di trasporto a *tcp*, infine il servizio, girerà sulla porta 30330. L'oggetto registrazione apparterrà ad una classe che implementa l'interfaccia *“RegisterListener”* e come già detto, conterrà metodi che consentono di interrogare l'esito dell'operazione di registrazione del servizio:

```
public void serviceRegistered(DNSSDRegistration
      registration, int flags, String serviceName, String
                              regType, String domain) {
   System.out.println("serviceRegistered()");
}

public void operationFailed(DNSSDService service,
                                    int errorCode) {
   System.out.println("operationFailed()");
}
```

Il primo metodo segnala l'avvenuta registrazione del servizio, il secondo riguarda il fallimento. Nel momento in cui il servizio viene registrato e reso pubblico, viene richiamato il metodo *serviceRegistered()*. Esiste anche un altro modo di pubblicare il servizio, più completo. Infatti oltre al semplice nome del servizio possiamo pubblicare tutta una serie di informazioni che vengono registrate in un oggetto *TXTRecord*

```
TXTRecord txtRecord;
txtRecord = new TXTRecord();
txtRecord.set("Nome","Foffo");
txtRecord.set("OS","Windows");
registration=DNSSD.register(DNSSD.NO_AUTO_RENAME,
         0,"iop2p","_iop2p._tcp", null, null, 30333,
                           txtRecord, registrazione);
```

Prima di tutto, abbiamo creato una nuova istanza della classe *TXTRecord*. Successivamente, abbiamo settato i vari attributi dell'oggetto, tramite il metodo *set()*. Infine, abbiamo registrato il servizio. In questo caso, abbiamo utilizzato un costruttore che prende come parametro anche il *TXTRecord* appena creato. In questo modo, il nostro servizio sarà raggiungibile esattamente come nel caso precedente prima ma in più, esporrà alcune informazioni aggiuntive.
Noterete anche, i due parametri settati a *“null”*, rispettivamente indicano il “dominio” e il nome dell'host. In particolare, il “dominio” è una stringa che potrà essere utilizzata in seguito per scopi specifici. Ad esempio, per velocizzare la ricerca di un servizio, potrebbe essere possibile restringerla a un particolare dominio piuttosto che a un'intera rete.

DISCOVERY DEI SERVIZI

Il primo passo è creare un elenco di “domini raggiungibili” dal nostro computer. In sostanza, il software andrà a spasso per la rete e recupererà tutti i domini esistenti, salvandoli poi in una lista. Per questa operazione, creeremo una classe che implementerà l'interfaccia *DomainListener*

```
class ModelloDomini implements DomainListener {
   Vector dati;

   public ModelloDomini() {
      dati=new Vector();
   }
   // Questo metodo viene richiamato
   // quando viene trovato un dominio
   // In questo caso memorizziamo tutti
   // i domini presenti in Vector

   public void domainFound( DNSSDService
                  domainEnum, int flags, int ifIndex,
                                   String domain) {
      if ( !dati.contains( domain))
         dati.addElement( domain);
      System.out.println("domainFound()");
      System.out.println("domain "+domain);
   }
```

```
  //Come il metodo precedente
  // anche questo viene richiamato
  // quando varia l'insieme di
  //domini raggiungibili. In questo
  // caso viene segnalata la perdita di un
  // dominio (senza specificare le cause)

  public void domainLost( DNSSDService domainEnum,
                  int flags, int ifIndex, String domain) {
    if ( dati.contains( domain))
        dati.removeElement( domain);
    System.out.println("domainLost()");
    System.out.println("domain "+domain);
  }
  public void operationFailed( DNSSDService service,
                                          int errorCode) {
    // Qui viene gestita il fallimento di
    // un'operazione
    // Viene anche passato il codice di
    // errore del quale possiamo trovare
    // il significato
    // nella documentazione di Bonjour
  }
}
```

A questo punto, possiamo richiamare il metodo della classe *DNSSD*, che ci consente di enumerare i domini trovati

```
domainList=new ModelloDomini();
domainBrowser = DNSSD.enumerateDomains(
           DNSSD.BROWSE_DOMAINS, 0, domainList);
```

In questo modo avremo memorizzato nell'istanza di *ModelloDomini* tutte le informazioni che il DNSSD riuscirà a trovare. Il passo successivo è quello riguardante l'implementazione dell'interfaccia *BrowseListener*. Inizieremo, prima cercando tutti servizi basati su DNSSD, poi raffinando la nostra ricerca. Per fare ciò, definiremo una classe che richiameremo più volte definendo il grado di profondità della nostra ricerca.

```
class ServiceListener implements BrowseListener {
  Risolver risolver = new Risolver();
  DNSSDService resolver;
  Ricerca ricerca;
  int tipo;
  public ServiceListener(Ricerca ricerca,int tipo) {
    this.ricerca=ricerca;
    this.tipo=tipo;
  }
  public void operationFailed( DNSSDService service,
                                         int  errorCode) {
    System.out.println("operationFailed()");
  }
  public void serviceFound( DNSSDService browser,
        int flags, int ifIndex, String serviceName, String
                              regType, String domain) {
    System.out.println("serviceFound() "+tipo);
    try {
      if (tipo==1)
        regType = serviceName + (
                        regType.startsWith( "_udp.")
                                 ? "._udp." : "._tcp.");
      if (tipo==1)
    ricerca.serviceDiscovered(
           flags,ifIndex,serviceName,regType,domain);
    else if (tipo==2) {
        System.out.println(flags+" "+ifIndex+"
          "+serviceName+" "+regType+" "+domain);
       if (serviceName.equals("iop2p"))
        resolver=DNSSD.resolve(0,ifIndex,serviceName,
                             regType,domain,risolver);
      }
   } catch(Exception e){
     e.printStackTrace();
   }
}
public void serviceLost( DNSSDService browser, int
        flags, int ifIndex, String serviceName, String
                              regType, String domain) {
    System.out.println("serviceLost(): Il servizio
                   "+serviceName+" e' stato perso");
  }
}
```

Abbiamo implementato i metodi che vengono richiesti dall'interfaccia *BrowseListener*. Il metodo più importante è *serviceFound()*, che appunto viene richiamato nel momento in cui viene trovato un servizio. In questo metodo, viene definito un comportamento per la ricerca dei servizi basati su DNSSD (ovvero quando la variabile *int* tipo è uguale a 1) e uno per la ricerca di altri tipi di servizio.
Dare il via alla ricerca adesso è abbastanza semplice

```
ServiceListener serviceListener=new
                          ServiceListener(ricerca,1);
DNSSDService servicesBrowser = DNSSD.browse(0,
     0, "_services._dns-sd._udp.", "", serviceListener);
```

Il sistema *Bonjour* quindi, notificherà all'oggetto *serviceListener* tutti i servizi trovati. Questo oggetto, a sua volta, richiamerà l'oggetto ricerca (istanza della classe d'esempio che abbiamo utilizzato) per notificare l'esito positivo.
Nel metodo *serviceDiscovered()*, della classe *Ricerca* chiameremo, per la seconda volta, il metodo *browse()* di DNSSD, questa volta andando a cercare uno specifico servizio. Infatti, adesso passiamo come parametro al *Service-*

Listener l'*int 2*, che ci permetterà di sapere se esistono servizi con il nome *"iop2p"*.

```
DNSSDService serviceBrowser;
serviceListener=new ServiceListener(this,2);
serviceBrowser = DNSSD.browse( 0, 0, regType, "",
                                        serviceListener);
```

Se viene trovato un servizio, con il nome che abbiamo specificato, si passa alla richiesta di informazioni dettagliate su questo servizio. Anche in questo caso, al metodo *resolve()*, dovremo passare un'istanza di un'ulteriore classe che implementa l'interfaccia *ResolveListener*.

```
resolver=DNSSD.resolve(0,
     ifIndex,serviceName,regType,domain,risolver);
```

L'oggetto risolver, implementa l'interfaccia *ResolveListener*. Il metodo che sarà richiamato in questo caso è *serviceResolved()*, del quale potete vedere, qui di seguito, l'implementazione che ci permette di recuperare le informazioni sull'host su cui sta funzionando il servizio *iop2p*, la porta e il descrittore *TXTRecord* (se presente).

```
public void serviceResolved(DNSSDService resolver,
              int flags, int ifIndex,String fullName,String
             hostName, int port, TXTRecord txtRecord) {
   System.out.println("serviceResolved()");
   System.out.println("Hostname "+hostName);
     if (txtRecord.contains("Nome"))
     System.out.println(txtRecord.
                             getValueAsString("Nome"));
}
```

Tutto questo rimbalzare di interfacce e metodi, non è il massimo della chiarezza, ma dopo averne capito il funzionamento, possiamo anche creare una classe wrapper, che permetta di gestire tutte le funzionalità di ricerca, senza dover implementare ogni volta cosi tante classi. Nelle **Figure 1** e **2** accanto, potete trovare uno schema riassuntivo del funzionamento di queste interazioni.

MAPPA DELLE CONDIVISIONI

Il nostro lavoro con Bonjour è quasi terminato. Ciascun PC, espone il manifesto del servizio e ciascun pc è in grado di scovare altri computer che espongono lo stesso servizio. Non ci resta che implementare il servizio di file sharing. Ovviamente, ciascun computer

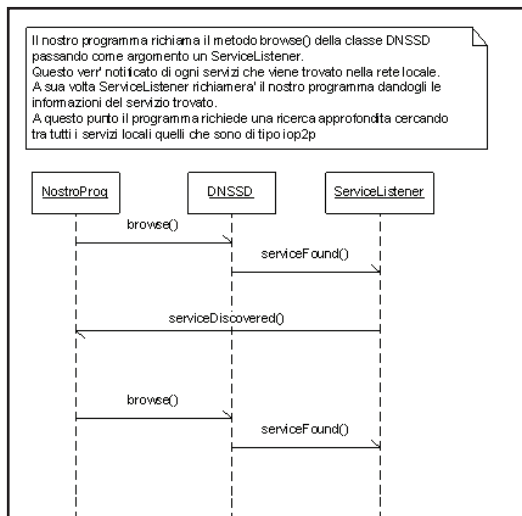



Fig. 1: *: L'interazione tra le classi per la ricerca dei generici servizi*

dovrà essere contemporaneamente client e server, di fatto dovrà poter downloadare i file dagli altri computer e consentire agli altri di scaricare file dal proprio sistema locale.
Partendo dal presupposto che, la mappa delle nostre condivisioni, sarebbe troppo grande per entrare nel *TXTRecord*, creiamo un server che distribuisce la mappa per ogni servizio *iop2p*. Quando troviamo un servizio *iop2p*, con numero di porta 30330 il server della mappa, sarà disponibile sulla porta 30331 (30330+1) dello stesso host.
Prima di tutto dobbiamo dare origine ad una classe che crei una struttura dati da pubblicare.

```
public class ListMantainer {
   private String basedir;
   private String separatore;
   private String map;
   private Hashtable ht;
   private int quanti=0;

   public ListMantainer(String basedir) {
      this.basedir=basedir;
      this.separatore=File.separator;
      this.map="";
      this.ht=new Hashtable();
   }
   public String getMap() {
      return map;
   }

   public void createMap() {
      File f=new File(basedir);
      createNode(f);
   }

   public void createNode(File f) {
      try {
```

```
        File[] filePresenti=f.listFiles();
        for (int i=0;i<filePresenti.length;i++) {
            if (filePresenti[i].isDirectory()) {
              createNode(filePresenti[i]);
            } else {
              String
              hash=createHash(filePresenti[i].
                                       getCanonicalPath());
              map=map+filePresenti[i].
                                   getName()+"#"+hash+"#";
              ht.put(hash,filePresenti[i].
                                       getCanonicalPath());
            }
        }
      } catch(Exception e) {
          e.printStackTrace();
      }
  }

  public static void main(String a[]) {
      ListMantainer lm=new ListMantainer(".");
      lm.createMap();
  }
  public String getPath(String hash) {
      return (String)ht.get(hash);
  }
  public String createHash(String value) {
      try {
        Security.addProvider(new
                       com.sun.crypto.provider.SunJCE());
        MessageDigest sha =
        MessageDigest.getInstance("SHA-1");
        sha.update(value.getBytes());
        byte[] hash = sha.digest();
        String hashString=new String();
        for (int i=0;i<hash.length;i++)
            hashString=hashString+hash[i];
        return hashString;
      } catch(Exception e)
      {
        e.printStackTrace();
        return null;}
  }
}
```

Un oggetto di questa classe ciclerà sulla directory passata al costruttore e controllerà il contenuto della condivisioni. Per creare la mappa dovremo richiamare il metodo *createMap()* e successivamente il metodo *getMap()*.
Dal punto di vista dell'utente che si collega distribuiremo una lista di file e il rispettivo hash calcolato con *SHA-1*. Quando un utente richiederà un file ne passerà l'hash e grazie ad un *Hashtable* sapremo subito dove si trova il file il file in questione. Per quanto riguarda la distribuzione della lista ecco il semplice server che dovremo avviare

```
class IoP2PServerList extends Thread{
  boolean running;
  public IoP2PServerList() {
    running=true;
  }
  public void run() {
     try{
     ServerSocket ss=new ServerSocket(30331);
     while(running) {
        Socket s=ss.accept();
        ListMantainer lm=new ListMantainer(".");
        lm.createMap();
        String map=lm.getMap();
        ObjectOutputStream oos=new
        ObjectOutputStream(s.getOutputStream());
        oos.writeObject(map);
        oos.flush();
        oos.close();
     }
   }  catch(Exception e) {}
  }
}
```

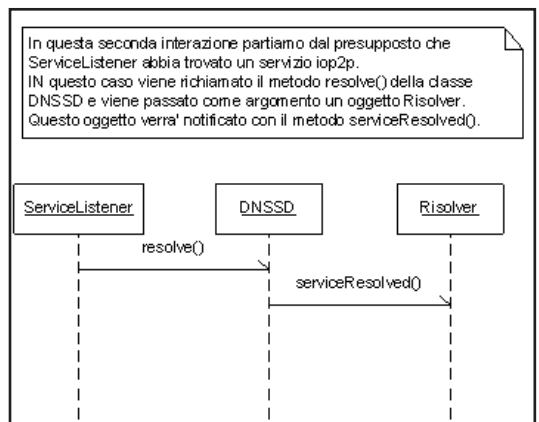



Fig. 2: ***:L'ultima interazione che ci permette info dettagliate sul servizio iop2p***

TRASFERIMENTO FILE

Ora scriviamo un semplice server per trasferire il file richiesto. Sappiamo già, che riceveremo l'hash del file che il client vuole scaricare. Quindi, diamo origine prima di tutto ad un server multithread che crea un Thread per ogni richiesta.

```
class IoP2PServer extends Thread{
  boolean running;
  public IoP2PServer() {
    running=true;
  }
  public void run() {
    try{
      ServerSocket ss=new ServerSocket(30330)
                                   while(running) {
```

SUL WEB

Esistono anche altre librerie/framework che permettono la definizione di servizi come Bonjour. Chiaramente cambia il livello di libertà nella definizione, perchè mentre Bonjour permette di pubblicare un manifesto del servizio e lascia poi tutta l'implementazione allo sviluppatore, altri framework decidono quali interfacce deve implementare il nostro servizio.
http://www.jini.org/
http://www.sun.com/software/jini/
http://www.jxta.org/
http://www.sun.com/jxta/

```
            Socket s=ss.accept();
            IoP2PServant servant=new
            IoP2PServant(s);
            servant.start();
        }
    } catch(Exception e) {}
    }
}
```

Per ogni connessione che arriva, faremo partire un *IoP2PServant*, che gestisce tutta la comunicazione con il client.
Per gestire cosa ci verrà passato dal client, abbiamo semplicemente istanziato un *ObjectInputStream* e un *ObjectOutputStream*, trasferendo quindi veri e propri oggetti.

```
class IoP2PServant extends Thread{
    Socket s;
    ObjectInputStream ois;
    ObjectOutputStream oos;

    public IoP2PServant(Socket s)
    {
        try {
            this.s=s;
            this.ois=new ObjectInputStream(
                                    s.getInputStream());
            this.oos=new ObjectOutputStream(
                                    s.getOutputStream());
        }
        catch(Exception e) {
        }
    }
    public void run()
    {
        try{
            String hash=(String)ois.readObject();
            ListMantainer lm=new ListMantainer(".");
            lm.createMap();
            String path=lm.getPath(hash);
            File f=new File(path);
            oos.writeObject(f);
            oos.flush();
        } catch(Exception e) { }
    }
}
```

Grazie al *ListMantainer*, che abbiamo implementato prima per avere la mappa delle nostre condivisioni, possiamo sapere subito il path del file all'interno del nostro harddisk, caricarlo in un oggetto *File* e serializzarlo via Socket.
Dal punto di vista del client, invece dovremo semplicemente richiedere il file, inviando con un *ObjectOutputStream* la stringa di hash e poi ricevere il File e salvarlo su filesystem.

```
Socket s=new Socket(ip,30330);
ObjectOutputStream oos=new
              ObjectOutputStream(s.getOutputStream());
ObjectInputStream ois=new
                 ObjectInputStream(s.getInputStream());
oos.writeObject(hash);
oos.flush();
File f=(File)ois.readObject();
FileInputStream fis=new FileInputStream(f);
File f_out=new File(f.getName());
FileOutputStream fos=new FileOutputStream(f_out);
while(fis.available()>0)
{
  fos.write(fis.read());
}
```

INTERFACCIA GRAFICA

L'ultima parte del nostro lavoro è visualizzare le informazioni riguardanti i file remoti. Per fare ciò, creeremo un semplice *JFrame* all'interno, del quale utilizzeremo una *JTable*.
Le informazioni che andranno a riempire la nostra tabella, saranno chiaramente parsate dal file delle condivisioni, che otteniamo via Socket.
Infine, dovremo semplicemente implementare un *Listener* sulla tabella, in modo tale che venga avviato il metodo che scarica il File e lo salva nella directory locale.

CONCLUSIONI

Il programma che trovate allegato alla rivista, permette lo scambio di file nella modalità descritta nell'articolo. Per avviare il server, dobbiamo far partire da riga di comando la classe *Registrazione* che mapperà tutti file presenti nella cartella di esecuzione. Il client per cercare i server *iop2p* è invece contenuto nel file *Ricerca.java*, il quale, permette anche un'interazione grafica per scegliere i file da scaricare.
Per quanto riguarda il programma che abbiamo realizzato, i miglioramenti per renderlo un vero e proprio P2P sono tantissimi: una migliore GUI, maggiori opzioni per l'utente, come la ricerca per nome e il settaggio delle cartelle da condividere. Ciò nonostante, abbiamo potuto capire come questa libreria della Apple, possa permetterci di creare degli interessantissimi servizi distribuiti. Una volta che, abbiamo reso disponibile la descrizione del servizio, dobbiamo soltanto preoccuparci della vera e propria implementazione. Insomma, un'arma in più per noi programmatori.

Federico Paparoni

L'autore, Federico Paparoni, può essere contattato per suggerimenti o delucidazioni all'indirizzo email federico.paparoni@javastaff.com <mailto:federico.paparoni@javastaff.com>

Effetto vetro per le finestre

Inpariamo come accedere alle API di Windows e sfruttare l'accesso al sistema operativo per creare applicazioni dotate di finestre trasparenti. Un effetto decisamente insolito per Java

Il valore di un'applicazione non dipende solo dalla qualità e dalle quantità delle funzioni che esporta, ma anche dalla facilità d'uso dell'interfaccia e dal suo aspetto estetico. Soprattutto in fatto di interfacce dalla grafica accattivante gli ultimi anni hanno visto nascere software frutto del genio di designer moderni e preparati. Così software dotati di finestre grafiche trasparenti o di forme particolarmente curate sono ormai all'ordine del giorno. Da tutto questo è rimasto fuori Java che, per la sua natura di linguaggio multipiattaforma, per il fatto di produrre un bytecode invece di un codice compilato e per la limitazione di non potere accedere alle API di sistema pena la perdita della sua caratteristica di portabilità, paga in termini di personalizzazione delle interfacce.
Tutto questo è vero solo in parte. Di fatto, in questo articolo illustreremo un metodo per accedere alle API di sistema e di conseguenza produrremo un'interfaccia Java dotata di form trasparenti. Il nostro codice funzionerà solo in ambiente Windows, ma in ogni caso se vorrete portare il tutto sotto Linux, perderete la bellezza delle form trasparenti ma dovrete riaggiustare solo poche linee di codice.

Fig. 1: Ecco come appare la nostra applicazione in esecuzione

I METODI NATIVI

Qualunque programmatore Java si sarà sicuramente imbattuto nella parola native tra gli identificatori che caratterizzano la firma di un metodo. Gli sviluppatori della Sun non hanno del tutto eliminato la possibilità di accedere alle API specifiche del sistema operativo ma hanno ben pensato di consentirne l'accesso laddove se ne riscontrasse la necessità. L'idea è semplice e ne vedremo un'implementazione pratica qualche paragrafo più in la. Utilizzeremo un comando java che genera un file header speciale in C++, ricompileremo il tutto in C++ ottenendo una libreria utilizzabile in Java tramite JNI.
Questo ci metterà a disposizione una serie di metodi, in particolare saremo in grado di utilizzare una libreria esterna: SkinLF, gratuitamente scaricabile dal sito *http://www.l2fprod.com* che farà da ponte tra java e la DLL ottenuta dal metodo precedentemente descritto. La libreria analizza un'immagine in ingresso e trasferisce le informazioni ai suoi metodi nativi, ricavati dalle API di Windows. È preferibile utilizzare immagini opache, che abbiano contorni ben definiti e con lo sfondo completamente trasparente. Prima di procedere, creiamo quindi un'immagine con tali caratteristiche oppure modifichiamone una esistente. La maggior parte di programmi di fotoritocco in circolazione sono adatti per lo scopo. In **Figura 2** è rappresentata la skin creata con Adobe Photoshop che verrà utilizzata nel corso dell'esempio.

I comandi per la generazione dell'Header C++ che ci servirà per creare la DLL accettano come parametro il nome del file che contiene le classi che tentano di utilizzare le API. Nel nostro caso scriveremo le nostre classi in un file chiamato *GUI.java*. Daremo poi in pasto a javah il nostro *GUI.Java* e ne ricaveremo un header pronto per essere compilato.
All'interno di *GUI.java* svilupperemo una classe principale, chiamata GUI, ed una classe *MouseMotion*, che gestirà il trascinamento col mouse.

Fig. 2: ***Il background realizzato con Photoshop***

UNA FINESTRA INVECE DI UN FRAME

Chi ha familiarità con le Java Swing sa che il primo passo da compiere per scrivere un'interfaccia grafica è estendere la classe JFrame. Dal momento che operiamo ad un livello più basso dobbiamo andare un pochino indietro nella gerarchia delle classi. Una superclasse di JFrame è Window, tra le cui funzioni vi è quella di disegnare la finestra. Deriviamo da tale classe:

```
public class GUI extends Window
{
}
```

All'interno dell'immancabile metodo main avviamo l'interfaccia grafica:

```
public static void main(String argv[])
{
        GUI gui = new GUI();
}
```

Il codice nativo della libreria *SkinLF*, in ambiente Windows, selezionerà la finestra alla quale applicare la trasparenza facendo riferimento al suo nome. Associamo quindi la nostra classe ad un JFrame chiamato *"ioprogrammo"*.

```
protected static JFrame frameOwner = new
                                   JFrame("ioprogrammo");

public GUI()
{
   super(frameOwner);
   ...
}
```

Il costruttore *GUI()* conterrà tutte le istruzioni per applicare la trasparenza e per creare i componenti grafici. Innanzitutto diamo una dimensione alla nostra finestra e posizioniamola al centro dello schermo:

```
Dimension screenSize = Toolkit
                    .getDefaultToolkit().getScreenSize();
setBounds((screenSize.width-WIDTH)/2,(
         screenSize.height-HEIGHT)/2,WIDTH,HEIGHT);
```

WIDTH ed *HEIGHT* sono due costanti che definiscono larghezza e altezza. Inseriamo quindi la nostra immagine all'interno della finestra come se fosse uno sfondo. A tale scopo utilizziamo un *JLayeredPane*, in cui gli elementi aggiunti man mano vengono collocati uno sopra l'altro.

```
ImageIcon icon = new ImageIcon("background.png");
JLabel background = new JLabel(icon);
background.setSize(background.getPreferredSize());

JLayeredPane layer = new JLayeredPane();
layer.setPreferredSize(background.getPreferredSize());
layer.add(background, JLayeredPane.DEFAULT_LAYER);

setLayout(new BorderLayout());
add("Center", layer);
```

L'immagine è visualizzata tramite un semplice componente *JLabel*, aggiunto poi al *JLayeredPane*, unico pannello che risiede nella Window. Ogni altro elemento grafico deve essere aggiunto alla finestra non passando per il metodo *add()*, bensì per il layer:

```
layer.add(componente, JLayeredPane.PALETTE_LAYER);
```

Per applicare la trasparenza bisogna effettuare quattro operazioni:

1. Ottenere un'istanza della classe *NativeSkin*.
2. Creare una *Region* per ritagliare l'immagine.
3. Associare la *Region* alla finestra.
4. Rendere trasparente la finestra.

DOVE TROVARE JAVAH

Se non vogliamo digitare la locazione completa del comando *javah* possiamo aggiornare la variabile d'ambiente *PATH* con il percorso *j2sdk_home/bin.* Esempio:

```
set PATH=%PATH%;c:
        \programmi\java
          \jdk1.5.0\bin
```

Ecco identificate le classi principali appartenenti alla libreria *SkinLF*. Di *NativeSkin* ne conserveremo l'istanza in una variabile globale, volendo cambiare il valore di trasparenza in un secondo momento. Ecco il codice:

```
if (NativeSkin.isSupported())
{

   builder = NativeSkin.getInstance();
   Region region = builder.createRegion(
                              icon.getImage());
   builder.setWindowRegion(this, region, true);
   builder.setWindowTransparency(this,200);

}
else
   System.err.println("Piattaforma non supportata");
```

L'intero per definire la trasparenza è un valore compreso tra 0 e 255, range a cui appartiene il canale alpha. Per arricchire l'interfaccia sono stati aggiunti un bottone per la chiusura dell'applicazione e qualche etichetta di testo. Le possibilità sono ovviamente infinite. Teniamo sempre presente che l'inserimento di ulteriori elementi grafici deve avvenire tramite il pannello layer e non tramite la finestra stessa.

```
JLabel title = new JLabel("Esempio di trasparenza in
            JAVA",new ImageIcon("icon.png"),
                      SwingConstants.LEFT);
title.setBounds(30,10,400,64);
title.setFont(new Font(title.getFont().getName(),
                              Font.BOLD,16));
layer.add(title, JLayeredPane.PALETTE_LAYER);
```

LA FINESTRA IN UN TASCHINO

Usare la classe Window nella creazione di un'interfaccia grafica ha i suoi svantaggi. Uno di essi è non poter utilizzare la taskbar del sistema operativo per la visualizzazione del

proprio programma. Tale compito è infatti delegato alla classe Frame. Possiamo risolvere utilizzando la system tray, rimanendo in tema di metodi nativi. L'ottima libreria open source Systray4j è scaricabile al seguente indirizzo: http://systray.sourceforge.net/. Il suo utilizzo è molto semplice, quasi come l'implementazione dei classici menu. In figura possiamo ammirare la tecnica della trasparenza insieme all'uso della systray.

Sebbene possiamo dire di aver raggiunto il nostro obiettivo, dobbiamo riflettere su un ultimo punto. L'azione di trascinamento del mouse sulla finestra non sortirà alcun effetto in quanto è la classe JFrame che si preoccupa di tale operazione di alto livello. Implementiamola "from scratch":

```
mouseMotion = new MouseMotion();
addMouseMotionListener(mouseMotion);
addMouseListener(new MouseAdapter()
{
   public void mouseReleased(MouseEvent me)
   {
      if (!moveable) return;
      mouseMotion.clearPosition();
      builder.setWindowTransparency(singleton,200);
   }
   public void mousePressed(MouseEvent me)
   {
      if (!moveable) return;
      builder.setWindowTransparency(singleton,50);
   }
}
);
```

La classe *MouseMotion*, contenuta interamente nel file sorgente, deriva da *MouseMotionListener*. Il suo metodo *mouseDragged()* è richiamato quando l'utente cerca di spostare la finestra. La posizione è aggiornata invocando *setLocation()* con le coordinate contenute nell'evento *MouseEvent* d'ingresso.
Inoltre, per deliziare gli occhi, aumentiamo molto la trasparenza solo durante l'azione di trascinamento.
L'effetto che ne viene fuori è poco consono ad un programma java e sicuramente desterà non poco stupore. Inoltre su pc relativamente veloci non dovrebbero verificarsi particolari cali di prestazione. Per ultimare la creazione, visualizziamo il tutto ed aggiungiamo un tocco d'importanza.

```
   setVisible(true);
   alwaysOnTop("ioprogrammo",true);
```

GENERIAMO L'HEADER

Il nostro scopo sarà quello di utilizzare le API di Windows per conferire alla nostra finestra la peculiarità *"always on top"*. Prima cosa da fare è aggiungere il metodo nativo alla classe principale:

```
public native void alwaysOnTop(
                     String title, boolean ontop);
```

Come possiamo vedere, tale metodo non presenta un corpo ma solo un'intestazione. L'implementazione dovrà avvenire attraverso un linguaggio nativo, C++ per Windows nel nostro caso. Genereremo il file header tramite un apposito tool a corredo con la JDK.
Apriamo una console dei comandi e posizioniamoci nella directory di lavoro. Dopo aver compilato GUI.java digitiamo il seguente comando:

```
javah –jni GUI
```

Verrà prodotto il file *GUI.h* che useremo per creare la nostra DLL *(Dynamic Linked Library)*. Visual C++ per Windows consente la creazione di librerie dinamiche. Tutto ciò che dobbiamo fare è aggiungere una nuova funzione, con l'inclusione dei relativi header.
Ecco il codice:

```
#include <jni.h>
#include "GUI.h"
#include <windows.h>

JNIEXPORT void JNICALL Java_GUI_alwaysOnTop(
        JNIEnv *env, jobject, jstring title, jboolean flag)
{

    char buf[128];
    const char *str = env->GetStringUTFChars(title, 0);
    strcpy(buf, str);
    (env)->ReleaseStringUTFChars(title, str);

    HWND hwnd;
    hwnd=::FindWindow(NULL,buf);
    if (flag)
      SetWindowPos(hwnd,HWND_TOPMOST,0,0,0,0,
                          SWP_NOMOVE|SWP_NOSIZE);
    else SetWindowPos(hwnd,HWND_NOTOPMOST,
                   0,0,0,0,SWP_NOMOVE|SWP_NOSIZE);
    return;
}
```

Per comprenderne meglio l'intestazione bisognerebbe studiare il funzionamento delle librerie JNI *(Java Native Interface)*. In ogni caso possiamo aiutarci col prototipo di funzione contenuto nell'header *GUI.h*, generato automaticamente. Il codice sopra riportato ottiene il riferimento alla finestra Java conoscendone il nome, ed effettua la relativa chiamata a sistema. La documentazione relativa alle API è consultabile sul sito Microsoft. Compiliamo la DLL e copiamola nella directory di lavoro.
Quindi, carichiamola in un blocco static, primo ad essere letto durante la fase di caricamento del programma:

```
static
{
   try
   {
      System.loadLibrary("./nativeskinwin32JAWT");
      System.loadLibrary("./AlwaysOnTop");
   }
   catch(Throwable t) { t.printStackTrace(); }
}
```

I JAVA LOOK AND FEEL

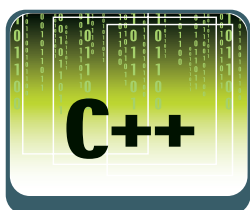
Il metodo più veloce per conferire personalità al proprio programma java è sicuramente quello di cambiare il Look And Feel. Un LNF è un insieme di codice e texture che modificano l'aspetto di tutti i componenti grafici. I bottoni, i campi per l'inserimento, i menu e i dialog cambiano forma e colore. Possiamo far coesistere una veste allegra ed una professionale all'interno dello stesso software. E tutto con pochissimo sforzo. Basta effettuare una ricerca in rete e scaricare il LNF più adatto alle proprie esigenze. Un buon punto di partenza è il sito http://www.javootoo.com, contenente i link agli stili più utilizzati. Inoltre, creare un LNF diventa più semplice con la versione 5 di java, consentendo anche ai meno esperti di personalizzare le proprie creazioni.

CONCLUSIONI

Con pochi semplici passi abbiamo creato effetti grafici di un certo rilievo. Adesso siamo in grado di arricchire le nostre applicazioni con qualcosa che possa veramente colpire l'utente finale. Risultato notevole se pensiamo alle limitazioni di java. Magari usando un particolare look and feel possiamo abbellire i componenti interni e, perché no, progettare un'interfaccia che permetta l'uso di più skin.
Occhio a non esagerare però. I processori a 512 bit non sono stati ancora inventati (o almeno credo).

Antonio Trapani

Terrain Mapping gioco o realtà?

Impariamo a realizzare ambienti virtuali che simulano gli spazi aperti. Fra boschi, montagne, terreni sconfinati, utilizzeremo il computer per realizzare ogni nostra fantasia!

Sono sempre di più i videogiochi in commercio che presentano una visuale su un terreno molto ampio. Basti pensare, ad esempio, ai titoli strategici in tempo reale, nei quali è possibile muovere diverse unità seguendole dall'alto a volo d'uccello, zoomare sul campo, spostarsi in maniera istantanea con la visuale ecc. In altre parole la possibilità di rappresentare una vasta area di gioco, è un requisito fondamentale per le meccaniche del gioco stesso; per cui, questa caratteristica, detta *"terrain rendering" (TR)*, deve essere implementata al meglio nel motore che muove il tutto.
Proprio per questa sua importanza e diffusione di utilizzo, il *terrain rendering* è molto spesso implementato in maniera nativa in molti motori grafici real-time 3D. Che vuol dire "maniera nativa"? A pensarci bene qualsiasi framework 3D potrebbe disegnare una superficie articolata: basterebbe caricare una mesh rappresentante il terreno, e il gioco sarebbe fatto! Questo discorso fila, ma fino a un certo punto. I problemi per questo approccio sono differenti, ad esempio:

- la mesh dovrebbe essere molto grande con conseguente impiego spropositato di memoria;
- difficilmente vedremo mai a schermo la mesh completa, ma solo una parte di essa: le risorse di calcolo impiegate per gestire la parte nascosta sarebbero sprecate;
- la costruzione stessa della mesh dovrebbe essere fatta "a mano" con un editor 3D, nonostante sia invece un'attività perfettamente automatizzabile, secondo alcuni parametri, magari anche a run-time;
- le coordinate di mapping delle texture sarebbero esageratamente piccole rispetto alle dimensioni della mesh stessa: altra memoria male utilizzata.

È quindi auspicabile, che ci sia un supporto diretto a questa tecnica, che consenta di effettuare in maniera automatica tutte le ottimizzazioni rese possibili dalle particolari caratteristiche del risultato che si deve ottenere.

L'APPROCCIO DI IRRLICHT

IrrLicht prevede la possibilità di effettuare il TR in maniera nativa, apportando tutte le ottimizzazioni del caso per rendere il disegno a schermo della scena più veloce possibile.
È possibile visualizzare un terreno, con tanto di texture, a partire da una *heightmap*. La *heightmap* (mappa di altezze) non è altro che un file grafico in toni di grigio. È come se vedessimo una mappa in miniatura del terreno che vogliamo visualizzare. Le zone più scure corrispondono a livelli più bassi del terreno, mentre quelle più chiare sono i rilievi.

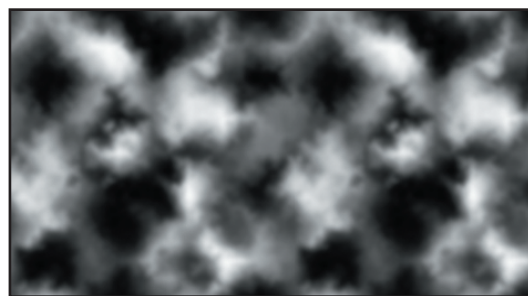



Fig. 1: Un esempio di HeightMap. Le zone più scure corrispondono a livelli bassi, quelle chiare a livelli alti

Dopo avere ricostruito il terreno a partire dalla *HeightMap*, è possibile applicarvi delle texture per rendere più reale l'effetto ottenuto.
Ad esempio, la texture che rappresenta un terreno verde, un deserto e un lago. Le texture vengono applicate per mezzo di un fattore di scala, che può adattarsi al livello di dettaglio che vogliamo ottenere. Ad esempio, se prevediamo inquadrature dall'alto, possiamo accontentarci di un livello "medio" di scalatura, mentre, per una scena vissuta in terza per-

sona, dovremmo, optare per un livello più elevato. L'algoritmo applicato da IrrLicht, per passare dalla *heightmap* all'effettiva visualizzazione finale, è abbastanza raffinato. Evita infatti, che ci si possa imbattere in forme che siano la trasposizione tout-court della heightmap (non vedremo i pixel trasformati in enormi cubi insomma!). Differenza di altezza fra zone vicine verranno trasformate in pendenze più o meno ripide che creano una continuità su tutto il terreno.

IL CODICE

Il codice di esempio si basa su quello disponibile all'interno del pacchetto completo di IrrLicht, tale codice può essere consultato per eventuali approfondimenti. Cominciamo con qualcosa di già noto ai lettori che abbiano seguito le precedenti puntate, ovvero l'inizializzazione standard di un semplice programma che usi IrrLicht:

```
#include <irrlicht.h>
using namespace irr;
#pragma comment(lib, "Irrlicht.lib")
#define WINDOW_TITLE L"Esempio di Terrain Rendering"
int main()
{ // Creo un device che utilizzi DirectX 9.0 come
                                              renderer
  IrrlichtDevice* device = createDevice(
          video::EDT_DIRECTX9, core::dimension2d
                                    <s32>(640, 480));
     if (device == 0)
        return 1; // impossibile creare il driver
  // Credo VideoDriver e SceneManager
  video::IVideoDriver* driver = device->getVideoDriver();
  scene::ISceneManager* smgr =
                           device->getSceneManager();
```

Abbiamo informato il compilatore del fatto che utilizzeremo il namespace *"irr"*, definito nell'header *"irrlicht.h"* e abbiamo consigliato al linker di utilizzare la libreria *"IrrLicht.lib"*. La funzione *main()* è l'unica funzione di questo semplice programma, al suo interno creiamo il device (utilizza DirectX 9.0 a una risoluzione di 640x480 pixel) il *VideoDriver* e lo *SceneManager*. Queste sono righe abbastanza standard che possiamo tranquillamente utilizzare mediante un comodo copia&incolla da un programma all'altro. Leggermente più specifiche, ma pur sempre standard, sono le istruzioni che seguono:

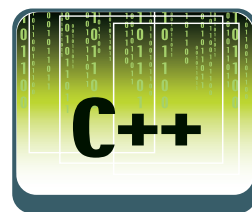
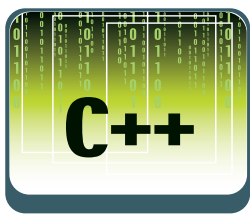
```
// Forzo la creazione di texture a 32 bit
driver->setTextureCreationFlag
  (video::ETCF_ALWAYS_32_BIT, true);
// Aggiungo e posiziono una videocamera
// controllabile coi tasti freccia
scene::ICameraSceneNode* camera =
  smgr->addCameraSceneNodeFPS(0,100.0f,1200.0f);
camera->setPosition(core::vector3df(
                         1900*2,255*2,3700*2));
camera->setTarget(core::vector3df(
                         2397*2,343*2,2700*2));
camera->setFarValue(12000.0f);
// Disabilito il cursore del mouse
device->getCursorControl()->setVisible(false);
```

Abbiamo forzato la creazione di texture a 32 bit (che non sempre sono di default!) e disabilitato il puntatore del mouse. È stata inoltre aggiunta una videocamera di tipo FPS che consente di muoversi all'interno della scena utilizzando il mouse e i tasti freccia, allo stesso modo di ciò che accade in qualsiasi videogioco sparatutto in prima persona.

NODO TERRENO

Il rendering di un terreno è implementato in IrrLicht tramite un particolare tipo di nodo chiamato *ITerrainSceneNode*. Per visualizzare un terreno quindi, è necessario creare un nodo di questo tipo, impostarlo correttamente con le opportune opzioni, e quindi aggiungerlo alla scena che si sta manipolando e che è gestita dallo *SceneManager*. L'impostazione avviene specificando la heightmap che sarà l'impronta del terreno, le texture da applicare (quella principale più la mappa di dettagli, che vedremo in seguito) e gli eventuali ridimensionamenti che diano al tutto un aspetto gradevole.
Ecco la traduzione in C++ di quello che abbiamo appena detto:

```
// Aggiungo un nodo TerrainSceneNode
scene::ITerrainSceneNode* terrain = smgr->
     addTerrainSceneNode("terrain-heightmap.bmp");
// Scalo la heightmap e disabilito l'illuminazione dinamica
terrain->setScale(core::vector3df(40, 4.4f, 40));
terrain->setMaterialFlag(video::EMF_LIGHTING, false);
// Imposto la texture
terrain->setMaterialTexture(0, driver->
                     getTexture("terrain-texture.jpg"));
```

La creazione a run-time della mesh che rappresenta il terreno, con le opportune ottimizzazioni, non ci esime dal considerare la mesh stessa al pari di qualsiasi altra mesh caricabile da file. Questo discorso vale anche per le diverse funzionalità applicabili a un corpo 3D. Tra queste funzionalità c'è la rilevazione delle collisioni. È possibile rilevare collisioni tra una mesh di terreno e un qualsiasi altro nodo di una scena nel consueto modo, utilizzando un *TriangleSelector* e il relativo animatore *(ISceneNodeAnimator)*. Quest'ultimo mette in relazione l'oggetto su cui ci stiamo concentrando con l'oggetto col quale desideriamo conoscere le eventuali collisioni.

Nel nostro caso, l'oggetto collidente è il nodo-videocamera creato prima: questo ci consentirà di spostarci sul terreno come se stessimo camminandoci sopra, senza oltrepassarlo.
Ecco il codice:

```
// Creo un TriangleSelector per la gestione delle collisioni
scene::ITriangleSelector* selector = smgr->
                createTerrainTriangleSelector(terrain, 0);
terrain->setTriangleSelector(selector);
selector->drop();
// Creo il corrispondente SceneNodeAnimator
// che risponda al contatto con la videocamera
scene::ISceneNodeAnimator* anim = smgr->
    createCollisionResponseAnimator(selector, camera,
    core::vector3df(60,100,60),core::vector3df(0,0,0),
                              core::vector3df(0,50,0));
camera->addAnimator(anim);
anim->drop();
```

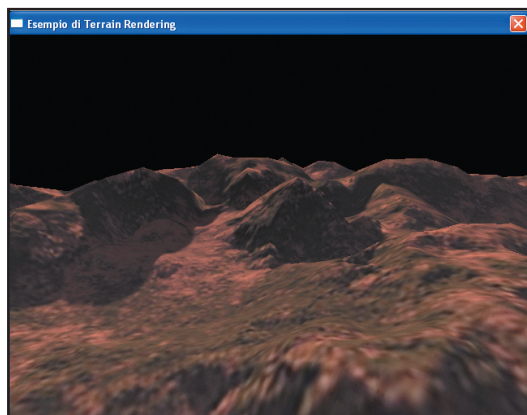



Fig. 2: ***Si noti come la definizione delle altezze è decisamente corretta, mentre il livello dei dettagli non è ancora ottimale***

Il programma è praticamente concluso! La parte principale è quella descritta in precedenza, non resta altro da fare che chiudere i giochi scrivendo il ciclo principale del programma, non prima però di avere opportunamente titolato la finestra di esecuzione:

```
// Titolo della finestra
device->setWindowCaption(WINDOW_TITLE);
// Ciclo principale del programma
while(device->run())
  if (device->isWindowActive())
  {
    driver->beginScene(true, true, 0 );
    smgr->drawAll();
    driver->endScene();
  }
  // Finalizzazione
  device->drop();
return 0;
}
```

MAPPA DEI DETTAGLI

Giocherellando col programma appena creato possiamo notare che la definizione del terreno è abbastanza soddisfacente, mentre le texture appaiono alquanto deludenti. Questo è dovuto al fatto che abbiamo usato un fattore di scala 1.0 (cioè non abbiamo scalato affatto!) e la texture del terreno è stata spalmata su una mesh alquanto grande. È possibile ovviare a questo inconveniente scalando di un fattore maggiore di 1.0 la texture. Questa, tuttavia, potrebbe non essere la soluzione migliore, in quanto la texture stessa potrebbe essere "legata" in qualche modo alla heightmap. Ad esempio, potrebbe essere colorata di blu laddove ci si aspetta ci sia uno specchio d'acqua. Aumentando la scalatura si avrebbero degli effetti poco gradevoli alla vista e sostanzialmente non corretti.

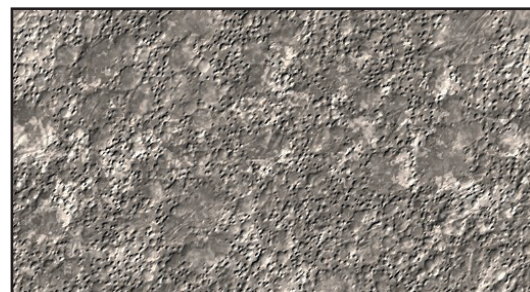



Fig. 3: ***Una Detail Map si sovrappone alla mappa principale aumentando il livello di definizione dei dettagli***

Un metodo migliore è quello di utilizzare una seconda mappa, detta *Detail Map* (mappa di dettagli) che si sovrappone alla texture principale aggiungendo però dei dettagli.
Per utilizzare la detail map è sufficiente modificare parte del codice precedente come segue:

```
// Imposto texture principale e mappa dei detttagli
terrain->setMaterialTexture
          (0, driver->getTexture("terrain-texture.jpg"));
terrain->setMaterialTexture
              (1, driver->getTexture("detailmap3.jpg"));
terrain->setMaterialType(video::EMT_DETAIL_MAP);
// La texture principale e' inserita una sola volta
// La mappa dei dettagli è inserita 20 volte
terrain->scaleTexture(1.0f, 20.0f);
```

Invocando il metodo *setMaterialTexture()* sulla posizione *1* (oltre che sulla posizione *0*) associamo al nodo terrain un'altra texture. Questa texture viene considerata da IrrLicht come una detail map, ma solo dopo avere impostato il parametro *video::EMT_DETAIL_MAP* tramite l'invocazione di *setMaterialType()*. Successivamente, effettuiamo una scalatura tramite la *scaleTexture():* manteniamo a 1.0 il fattore di scala della texture principale, mentre ripetiamo di un fattore 20.0 la texture dei dettagli. In questo modo questi saranno abbastanza precisi,

anche quando guarderemo molto da vicino il terreno renderizzato. Il risultato di questa ulteriore miglioria al codice è visibile in **Figura 4**.

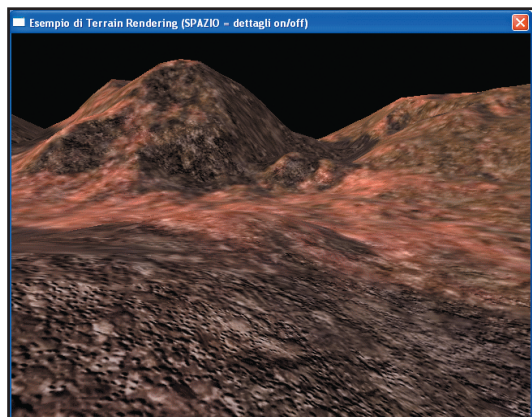



Fig. 4: ***Si noti come dopo l'applicazione della Detail Map, il livello di dettaglio è adesso decisamente migliore***

Per completare l'esempio, aggiungiamo la possibilità, da parte dell'utente, di modificare a run-time l'utilizzo o meno della detail map. Questo effetto è ottenuto impostando alternativamente i valori *video::EMT_DETAIL_MAP* e *video::EMT_SOLID* per il terreno tramite la *setMaterialType()*.
Per catturare la pressione del tasto che permette di effettuare questo cambiamento, (abbiamo scelto lo *SPAZIO*), dobbiamo derivare una classe dalla classe *IEventReceiver* di IrrLicht (una interfaccia in realtà). Il codice di questa classe è il seguente:

```
// L'EventReceiver che consente di visualizzare o
                                         meno i dettagli
class MyEventReceiver : public IEventReceiver
{ public:
  MyEventReceiver(scene::ISceneNode* terrain)
  {  // puntatore al nodo utilizzato
  Terrain = terrain; }
  bool OnEvent(SEvent event)
  { // check per la presisone del tasto SPAZIO
  if (event.EventType == irr::EET_KEY_INPUT_EVENT
                         && !event.KeyInput.PressedDown)
  { switch (event.KeyInput.Key)
    { // attiva/disattiva la detail map
      case irr::KEY_SPACE:
      Terrain->setMaterialType
       (Terrain->getMaterial(0).MaterialType
          == video::EMT_SOLID ? video::EMT_DETAIL_
                            MAP: video::EMT_SOLID);
       return true; } }
return false; }
private:
  scene::ISceneNode* Terrain;
};
```

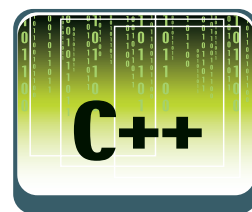
Come si può notare, nel caso il tasto premuto (e rilasciato!) sia *irr::KEY_SPACE*, e cioè lo *SPAZIO*, l'*EventReceiver* si occupa di effettuare il cambiamento tra i parametri visti in precedenza.
Per utilizzare *MyEventReceiver* è necessario inoltre aggiungere all'interno della funzione *main()* il seguente codice:

```
// Istanzio un oggetto MyEventReceiver in modo
// da poter abilitare la mappa di dettagli
MyEventReceiver receiver(terrain);
device->setEventReceiver(&receiver);
```

il codice completo e funzionante del programma che consente di cambiare in tempo reale tra rendering con e senza la detail map, è disponibile sul CD allegato.

CONCLUSIONI

Il terrain rendering è una tecnica a dir poco utile nella programmazione di diversi tipi di applicazione 3D. Data la vastità e la complessità della mesh che servirebbe per visualizzare un terreno realistico, è impossibile pensare che tutto ciò venga fatto "a mano" senza alcuna ottimizzazione da parte del motore, caricando una semplice mesh gigante. IrrLicht offre un supporto al terrain rendering, implementato attraverso l'uso di *heightmap* e *detail map*. La mesh che viene generata è ottimizzata a run-time in modo da evitare sprechi di risorse ma si comporta come una mesh "tradizionale" a tutti gli effetti. È ad esempio possibile applicare un animatore che rilevi le collisioni con un altro nodo della scena, come la videocamera utilizzata nei nostri esempi!

Alfredo Maroccelli

TERRENI IN VOXEL

Quando ancora i PC casalinghi non bruciavano MHz a ritmo così forsennato, tenere la struttura di una mesh molto grande in memoria era proibitivo e rappresentarla a schermo semplicemente un'utopia. Furono sviluppati allora diversi metodi alternativi, molto meno precisi. Uno di questi è il cosiddetto *Voxel Space*, cioè uno spazio nel quale il terreno viene disegnato per mezzo dei "*voxel*". Un *voxel* è l'equivalente tridimensionale di un pixel 2D. La tecnica è molto semplice, almeno nella sua versione base, e consentiva all'epoca di ottenere un risultato notevole. Gli ammassi di pixel verde-marroncino generati da questo tipo di algoritmi, infatti, erano anni luce avanti, come impatto grafico, alle montagne-piramidi senza texture degli arcaici motori 3D dei quali erano concorrenti. Molto diffusa tra i videogiochi di volo, ricordiamo come uno dei primi utilizzatori della tecnica il gioco *Comanche: Maximum Overkill*.

Guerra di Robot chi vincerà?

Divertente, emozionante, stimolante, creiamo in Java un automa e iscriviamolo a un moderno torneo a squadre pronto a sfidare migliaia di altri robot. Chi sarà il migliore?

C'è un'arena, ci sono due robots, e una serie di regole. Vince il robot che riduce l'altro ad una poltiglia di bit. È questo Jrobots, con la particolarità che i robot non esistono fisicamente. Si tratta piuttosto di due algoritmi che simulano il comportamento di due robot che combattono in un'arena. Due algoritmi a confronto, ciascuno operante con una propria logica per battere l'altro in una competizione basata su regole!
Semplice? Sì, ma la concorrenza è spietata!

L'ARENA

Il terreno di gioco è un quadrato da un km per lato. È importantissimo tenere a mente queste misure, dato che colpire le pareti dell'arena non è per nulla conveniente: la nostra velocità passa immediatamente a zero e perdiamo due punti vita su 100 totali. In questa arena, a seconda del tipo di partita, si affrontano due o più robot:

- **Single Match:** due robot combattono uno contro l'altro.
- **Double Match:** due coppie di robot combattono una contro l'altra.
- **Team Match:** quattro squadre di otto robot ciascuna combattono l'una contro l'altra.

È chiaro, che se non si gioca solo in Single Match, sarà necessario sviluppare anche un algoritmo in grado di riconoscere i nemici dagli amici, colpendo solo i primi. Qualsiasi sia la modalità, il tempo massimo per un match è di 180 secondi.

LE REGOLE

Il motore dei robot permette una velocità massima di 30 metri al secondo, ossia 100 km/h. Ovviamente questa velocità non è ragiunta istantaneamente, ma con un'accelerazione di 5 m/s. Per regolare la velocità possiamo agire sulla potenza, un valore percentuale che si imposta con una primitiva apposita. Se la potenza è al 0 %, il robot frenerà, con una decelerazione di 5 m/s. L'unica

COME INIZIARE

JRobots è un clone di CRobots, vecchio gioco scritto nel 1985 da Tom Poindexter. Leonardo Boselli lo ha ripreso, ricreato in Java e aggiornato all'era di Internet, trasformandolo in un interessantissimo Multiplayer. Il linguaggio in cui scrivere gli algoritmi per il nostro robot è Java, ma se ne avevate già preparato uno in C per CRobots la conversione è immediata, dopo tutto la sintassi è molto simile ed è strettamente vietato l'utilizzo di qualsiasi API Java. Per programmare il proprio robot è necessario scaricare, dal sito http://jrobots.sourceforge.net/, l'SDK che contiene l'emulatore dell'arena e le classi necessarie. Sullo stesso sito trovate la form per e procedere all'iscrizione ed effettuare l'upload della vostra creatura, sperando che non venga divorata in pochi secondi da rivali più esperti.

REQUISITI

Conoscenze richieste
J2SE

Software
J2SE 1.4.1 SDK o superiore

Impegno

Tempo di realizzazione

eccezione a questa regola è l'urto con le pareti, nel cui caso la velocità passa a 0 istantaneamente. I robot, sono equipaggiati con un cannone lancia missili. Non c'è un limite di munizioni, solo un ritardo di ricarica di un secondo fra un lancio e l'altro. Si può sparare in qualsiasi direzione. I missili viaggiano a 300 m/s, con una gittata massima di 700 metri, qualsiasi sia la velocità del robot al momento del lancio. Non è necessario che il missile colpisca direttamente il robot per danneggiarlo.
I punti danno arrecati, infatti, dipendono direttamente dalla distanza dell'esplosione:

- 5 metri 10 punti
- 20 metri 5 punti
- 40 metri 3 punti

I punti vita disponibili sono 100, esauriti i quali si viene distrutti.
È chiaro quindi che, prima di sparare bisogna ben controllare chi c'è nel raggio d'azione del missile: oltre al destinatario potrebbero esserci dei compagni di squadra, o addirittura noi stessi! Qui ci viene incontro lo scanner, attraverso il quale possiamo scoprire la presenza o meno di robot nelle vicinanze. Per essere precisi, con lo scanner possiamo analizzare il campo di battaglia, con un angolo di apertura da 1 a 20 gradi. Ad angoli più piccoli, corrispondono analisi più precise. Si ottiene però solo la distanza del robot più vicino, amico o nemico che sia.

FUNZIONI

Per muovere il robot abbiamo un unico metodo, *drive()* a cui passare la direzione (angolo) e la potenza erogata. Questo stesso metodo serve anche per frenare, se si imposta la potenza a 0.
Possiamo conoscere la posizione del nostro robot attraverso i metodi *loc_x()* e *loc_y()* che restituiscono le coordinate x e y a partire dall'angolo in alto a sinistra. Attraverso il metodo *speed()* invece otteniamo la velocità, in percentuale (*0%* sta per *0m/s, 100%* per *30 m/s*).
Per lanciare i missili è sufficiente un unico metodo, cannon(int degrees, int range) che per parametri prende l'angolo (a partire dalla posizione *"ore tre"*) e la gittata (distanza raggiunta la quale il missile esplode). La partita continua finché non rimane un solo robot in vita o scade il tempo massimo di 180 secondi. Sono utili quindi le due funzioni *damage()* e *time()*, che restituiscono rispettivamente i punti vita residui e il tempo trascorso in secondi dall'inizio della partita.

Anche per attivare lo scanner, basta una sola funzione, *scan(int degrees, int resolution)* alla quale, passando angolo di partenza e ampiezza dello scan da 1 a 20 gradi, ci fornisce la distanza del robot più vicino.
Il regolamento di JRobots vieta tassativamente, pena la non iscrizione del proprio robot alla lega, l'utilizzo di qualsiasi delle API Java. Dato che la fisica è alla base di questo gioco, fra i metodi forniti ci sono anche varie funzioni matematiche, che sopperiscono alla impossibilità di ricorrere alle librerie *java.lang.Math:*

- **int rand(int limit)** per generare numeri casuali.
- **int sqrt(int value)** radice quadrata.
- **int sin(int degrees)** funzione seno (il risultato è il seno per 100000).
- **int cos(int degrees)** funzione coseno (il risultato è il coseno per 100000).
- **int tan(int degrees)** funzione tangente (il risultato è la tangente per 100000).
- **int atan(int value)** funzione arcotangente (il valore deve essere 100000 volte quello reale ed il risultato è in gradi).
- **double d_sqrt(double value)** radice quadrata.
- **double d_sin(double radians)** funzione seno.
- **double d_cos(double radians)** funzione coseno.
- **double d_tan(double radians)** funzione tangente.
- **double d_atan(double value)** funzione arcotangente.
- **double deg2rad(double degrees)** per convertire un angolo da gradi a radianti.

IL ROBOT

Il robot è una semplice classe che estende la classe JJRobots e pubblica un metodo main eseguito all'inizio della partita. Tutte le altre classi da noi sviluppate, devono seguire un particolare formato nel nome:

```
__NomeClasse_
```

devono cioè iniziare con due underscore e terminare con uno. Non ci sono altre limitazioni, al di là della grandezza massima dei 20kb per classe.
Una delle problematiche principali è sicuramente legata al puntamento. Colpire un corpo in movimento è più complesso di colpirne uno fermo ovviamente. Se ci limitiamo ad individuare il nemico, con la funzione *scan()* e sparare verso quel punto, molto probabilmente non lo colpire-

mo. Questo perché nel tempo impiegato dal proiettile a raggiungere l'obiettivo, il nemico può spostarsi anche di molto. Sono fondamentali quindi algoritmi che prevedano la posizione dell'obiettivo, basandosi sulla sua velocità. Per calcolare la velocità bisogna conoscere la posizione del nemico in due momenti diversi, possibilmente anche distanti fra loro, dato che il metodo scan restituisce valori interi. Trovata la velocità del bersaglio, occorrerà rinfrescare la propria preparazione in Fisica per trovare la sua posizione nel momento dell'impatto. Sono utili, ovviamente, anche nozioni di calcolo vettoriale. Un altro punto "caldo" dello sviluppo è la scelta delle traiettorie da seguire nello spostarsi nell'arena. Un robot che si muove in linea retta fino al raggiungimento del bordo, cambiando direzione solo in quel momento, è un robot prevedibile e quindi un facile bersaglio.

UN ROBOT IN PRATICA

Analizziamo nel dettaglio il codice di un robot, Platoon, ovvero uno dei robot esemplificativi proposti da Leonardo Boselli nel suo sito. Il robot è composto da una sola classe. Fondamentale il nome, che deve iniziare con due underscore e terminare con uno, con tutti gli altri caratteri unicamente alfanumerici. Estendiamo la classe JJRobot. Definiamo le strutture dati, tutti interi e array di interi. La variabile *count* conterrà il numero di robot "amici" nell'arena, nelle variabili *targetX* e *targetY* invece troveremo la posizione dell'ultimo bersaglio individuato. In *LocX* e *LocY,* invece inseriremo le posizioni degli amici, evitando di colpirli accidentalmente. Nella prima parte del metodo main impostiamo id, variabile che identifica con un numero progressivo il nostro robot per riconoscere quale elemento della squadra sia. In questo modo, possiamo fare giocare in maniera diversa i vari elementi della squadra.

```
public class __Platoon_ extends JJRobot {
private static int count;
private static int[] cornerX = {50,950,950,50};
private static int[] cornerY = {50,50,950,950};
private static int targetX = 500;
private static int targetY = 500;
private static int locX[] = new int[8];
private static int locY[] = new int[8];
private static int corner1;
private int nCorner;
private int scan;
private int id;
void main() {
  if((id = id()) == 0) {
    count = 1;
    corner1 = rand(4);
  } else {
    count = id+1;
  }
```

Diamo un algoritmo che permetta al robot di muoversi in maniera "coerente". Il robot sceglie un angolo e si direziona verso quello. Nel tragitto spara qualche missile, tanto per gradire (primo switch). Poi quando si è avvicinato all'angolo scelto, rallenta fino al 50% della velocità, per non sbattere sulle pareti, continuando a sparare a ripetizione. Quindi cambia angolo e lo raggiunge, sempre sparando (secondo switch). Il movimento è prevedibile, ma abbastanza funzionale, specialmente nelle partite in team.

```
  nCorner = corner1;
  int dx = cornerX[nCorner]-(locX[id]=loc_x());
  int dy = cornerY[nCorner]-(locY[id]=loc_y());
  int angle;
  if(dx == 0) {
    angle = dy > 0? 90: 270;
  } else {
    angle = atan(dy*100000/dx);
  }
  if(dx < 0) angle += 180;
  drive(angle,100);
  switch(nCorner) {
    default:
    case 0: while(locX[id] > 150 || locY[id] > 150)
                                        fire2(); break;
    case 1: while(locX[id] < 850 || locY[id] > 150)
                                        fire2(); break;
    case 2: while(locX[id] < 850 || locY[id] < 850)
                                        fire2(); break;
    case 3: while(locX[id] > 150 || locY[id] < 850)
                                        fire2(); break;
  }
  do {
    drive(0,0);
    while(speed() >= 50) fire1();
    if(++nCorner == 4) nCorner = 0;
    dx = cornerX[nCorner]-loc_x();
    dy = cornerY[nCorner]-loc_y();
    if(dx == 0) {
      angle = dy > 0? 90: 270;
    } else {
      angle = atan(dy*100000/dx);
    }
    if(dx < 0) angle += 180;
    drive(angle,100);
    switch(nCorner) {
      default:
      case 0: while(locY[id] > 150) fire1(); break;
      case 1: while(locX[id] < 850) fire1(); break;
      case 2: while(locY[id] < 850) fire1(); break;
```

```
      case 3: while(locX[id] > 150) fire1(); break;
    }
  } while(true);
}
```

Individuare il nemico non è semplice. In questi due metodi il robot l'arena, cercando il nemico, nel primo solo nella zona opposta al muro in cui si trova, nel secondo invece a 360 gradi. La variabile *scan,* contiene la posizione angolare da cui iniziare l'analisi con il metodo *scan()*.

```
private void fire1() {
  switch(nCorner) {
    default:
    case 0: if(++scan > 470 || scan < 240) scan =
                                          250; break;
    case 1: if(++scan > 200 || scan < -30) scan =
                                          -20; break;
    case 2: if(++scan > 290 || scan <  60) scan =
                                           70; break;
    case 3: if(++scan > 380 || scan < 150) scan =
                                          160; break;
  }
  fire();
}
private void fire2() {
  if(++scan > 360) scan = 0;
  fire();
}
```

ecco il metodo che ci permette materialmente di trovare il nemico, utilizzando la funzione *scan()*. Nel primo *if* riconosciamo se abbiamo trovato un altro robot a portata di cannone. Se count è maggiore di 1, ci sono amici in gioco, non possiamo sparare quindi in maniera indiscriminata. Altrimenti abbiamo d'avanti un nemico… fuoco a volontà!

```
private void fire() {
  locX[id] = loc_x();
  locY[id] = loc_y();
  int range;

  if((range = scan(scan,1)) > 40 && range <= 740) {
    if (count > 1) {
      boolean shot = true;
      int shotX = locX[id]+range*cos(scan)/100000;
      int shotY = locY[id]+range*sin(scan)/100000;
      for(int ct = 0; ct < count; ct++) {
        .....
      }
    }
    else {
      cannon(scan,range);
      scan -= 10;
    }
```

Se abbiamo amici in gioco, non possiamo sparare senza controllare. Dobbiamo riconoscere il bersaglio e quindi saminare le posizioni dei nostri amici, contenute negli array *locX* e *locY*. Se nell'area intorno al bersaglio, con un raggio di 40 metri troviamo un amico, meglio non sparare.

```
    if(ct != id) {
      int dx = shotX-locX[ct];
      int dy = shotY-locY[ct];
      if(dx*dx+dy*dy < 1600) {
        shot = false;
        break;
      }
```

Se abbiamo settato a true la variabile booleana shot, il bersaglio è un nostro nemico. Possiamo utilizzare il metodo *cannon()* per lanciare il missile. Se invece è settata a *false*, meglio sparare all'ultimo bersaglio individuato!

```
  if(shot) {
      targetX = shotX;
      targetY = shotY;
      cannon(scan,range);
      scan -= 10;
    } else {
      int dx = targetX-locX[id];
      int dy = targetY-locY[id];
      int dist2 = dx*dx+dy*dy;
      if(dist2 > 1600 && dist2 <= 547600) {
        int angle;
        if(dx == 0) {
          angle = dy > 0? 90: 270;
        } else {
          angle = atan(dy*100000/dx);
          if(dx < 0) angle += 180;
        }
        cannon(angle,sqrt(dist2));
      }
    }
```

LE LEGHE

Attualmente ci sono due leghe, una dei veterani e l'altra dei cadetti. Si parte in serie *"B"*, ma se si è sufficientemente forti già in una sola stagione, si può essere promossi al grado di veterani.
Considerato che c'è un nuovo campionato ogni mese, non c'è il roschio di non aver modo di aspirare al titolo.
Il problema semmai è costituito dalla forza dei rivali molto più esperti di voi dato che il gioco esiste dal 1999 o ancora prima se si considera CRobots.
Ma non credo che questo vi spaventerà, giusto?

Luca Mattei

Effetti speciali sulle immagini

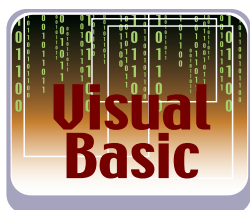
Non sempre la matematica è noiosa. In questo articolo vedremo come trasformare un'immagine, in base ai valori di una funzione, per ottenere fantastici e divertenti effetti speciali

Che cosa vuol dire applicare un effetto ad un'immagine? Sostanzialmente vuol dire prendere un'immagine base, applicare una trasformazione ai pixel che la compongono e ricopiare il risultato ottenuto su un'immagine di output. Va da se che la chiave per la realizzazione di filtri particolari sta nelle varie tipologie di trasformazioni applicate sui pixel. In sostanza è necessario applicare una funzione matematica a un pixel per ottenere il suo corrispondente trasformato secondo la funzione applicata.
In questo appuntamento descriveremo come creare un'effetto speciale su un'immagine utilizzando la funzione API *BitBlt* che permette di copiare una mappa di bit da un device context ad un altro in base a dei parametri di trasformazione.
La *BitBlt* dunque è indicata per copiare i punti di un'immagine da una PictureBox ad un'altra e per specificare la loro posizione in base ai valori assunti da una funzione matematica; proprio quello che ci siamo proposti di fare in questo articolo. Prima di procedere, conviene fare un breve ripasso di matematica. Ricordiamo che una funzione matematica mette in relazione due grandezze, per esempio la temperatura e l'altitudine. In linguaggio matematico si afferma che una grandezza y, ad esempio la temperatura, è funzione di un'altra grandezza x, per esempio l'altitudine, se per ogni valore di x la funzione determina in modo univoco un valore di y (come è noto più si sale di quota e più la temperatura diminuisce). Le funzioni matematiche che, principalmente, utilizzeremo nei nostri esempi, sono quelle trigonometriche che esprimono relazioni tra angoli *(x)* e lunghezze *(y)*.
Nell'articolo dunque, dopo un breve cenno sulle funzioni trigonometriche, presenteremo un'applicazione che, grazie alla *BitBlt*, permette di trasformare un'immagine sulla base di una o più funzioni.

Fig. 1: La form che permette di disegnare le funzioni

TRACCIARE LE FUNZIONI TRIGONOMETRICHE

Le funzioni trigonometriche, come accennato, mettono in relazione degli angoli, misurati in gradi o in radianti, con delle lunghezze. Le funzioni trigonometriche principali sono seno, coseno e tangente che in Visual Basic diventano: *Sin(x)*, *Cos(x)* e *Tan(x)*.
Se per esempio consideriamo la funzione *y= Sin(x)*, questa restituisce un valore *Double (y)* che specifica la lunghezza del seno dell'angolo *x* espresso in radianti. I valori di *Sin(x)*, e quindi di *y*, sono compresi tra *-1* e *+1* e si ripetono (sono periodici) ogni *2*π*, cioè quando $x=1$ oppure $x=1+K*2*\pi$ con $K=1\ldots$ infinito, la *y* assume sempre lo stesso valore.
Ricordiamo che la trasformazione da gradi in radianti e viceversa è molto semplice: se *x* è

un angolo in gradi, il suo valore in radianti si ottiene moltiplicandolo per 6.28/360; viceversa un valore in radianti va moltiplicato per 360/6.28 per ottenere l'angolo in gradi. Inoltre si rammenta che 360 è il numero di gradi di una circonferenza, mentre 6.28, conosciuto anche come *2*π (2 per PiGreco)*, è il numero di radianti della stessa. Come capiremo negli esempi, la periodicità ed il fatto che i valori di *y* sono compresi tra *-1* e *+1*, ci permetterà di ottenere effetti grafici molto interessanti.
Di seguito illustriamo una routine che permette di tracciare il grafico delle tre funzioni trigonometriche principali; naturalmente, facendo le opportune modifiche, la routine può essere utilizzata per qualsiasi funzione.
Nell'esempio, i valori delle funzioni sono disegnati su una PictureBox con la proprietà *Pset*. Gli assi del piano cartesiano, in pratica l'asse orizzontale o asse *X* e l'asse verticale o asse *Y*, sono disegnati con la proprietà *Line*.
Create un nuovo progetto e sulla *form1* inserite una PictureBox, nominata *PicFun*, e un pulsante, nominato *PulsFun*.
Nella parte dichiarativa della form inserite le variabili *Passo* (Step), *Cx* e *Cy*. *Passo* verrà utilizzata per impostare la risoluzione dei punti del grafico, *Cx* e *Cy*, invece, conterranno le coordinate del punto centrale degli assi cartesiano, che nel nostro caso è il centro della PictureBox.

```
Dim Passo As Double
Dim Cx, Cy As Double
```

Il *Passo* viene impostato nella *Form_Load*.

```
Private Sub Form_Load()
 Passo = 0.001
End Sub
```

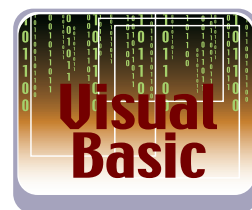
Per disegnare gli assi cartesiani possiamo usare la funzione *Assi* presentata sotto.

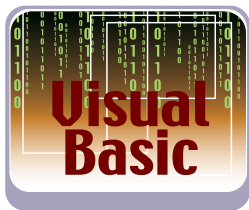
```
Private Sub Assi()
 Cy = PicFun.ScaleHeight / 2
 Cx = PicFun.ScaleWidth / 2
 PicFun.Line (0, Cy)-(PicFun.ScaleWidth, Cy)

`asse X
 PicFun.Line (Cx, 0)-(Cx, PicFun.ScaleHeight)

 `asse Y
End Sub
```

Nella Assi, dopo aver calcolato il centro della PicturBox, vengono tracciate due linee *(Line)* perpendicolari che si incrociano nel centro di coordinate *[Cx, Cy]*.
Siamo pronti per tracciare la funzione trigonometrica *seno*.
Nella procedura del pulsante *PulsFun* predisponiamo un ciclo che considera tutti i valori dell'asse *X*, con lo step impostato sulla variabile *Passo;* e in questo ciclo predisponiamo le istruzioni che valutano la funzione *Sin*, in base a *X*, e disegnano i punti sulla PictureBox.

```
Private Sub PulsFun_Click()
 Dim X, Y, X1, Y1 As Double
 For X = -(PicFun.ScaleWidth / 2) _
  To PicFun.ScaleWidth / 2 Step Passo
   Y = (PicFun.ScaleHeight * 0.25) * Sin(X)
   X1 = Cx + X
   Y1 = Cy - Y
   PicFun.PSet (X1, Y1), RGB(255, 0,0 )
 Next X
End Sub
```

Per aumentare l'ampiezza del seno, moltiplichiamo la funzione per il *25%* di *PicFun.ScaleHeight;* che i punti da disegnare sono calcolati rispetto al centro della *PicFun* e che essi sono impostati di colore *Rosso* con *RGB(255, 0,0)*. Un'ulteriore evoluzione è la procedura che permette di tracciare, contemporaneamente, in colori diversi, il *Sin*, il *Cos* e la *Tan*.

```
Private Sub PulsFun_Click()
 Dim X, Y, X1, Y1 As Double
 Dim r, g, b, fun As Integer
 Assi
 For fun = 1 To 3
  For X = -(PicFun.ScaleWidth / 2) _
  To PicFun.ScaleWidth / 2 Step Passo
   Select Case fun
    Case 1
     Y = (PicFun.ScaleHeight * 0.25) * Sin(X)
     r = 0
     g = 0
     b = 255
    Case 2
     Y = (PicFun.ScaleHeight * 0.25) * Cos(X)
     r = 0
     g = 255
     b = 0
    Case 3
     Y = Tan(X)
     r = 255
     g = 0
     b = 0
   End Select
   X1 = Cx + X
   Y1 = Cy - Y
```

LA LIBRERIA MSFORM2

La *MsForm2 library (FM20.dll)* contiene i controlli che ampliano le caratteristiche di alcuni elementi nativi di Visual Basic come il *Frame*, il *CommandButton*, l'*image* ecc. Per esempio sia il *Frame* che il *CommandButton* di *MsForm2* possono essere riempiti con un'immagine o con un colore. I Controlli di *MsForm2*, rispetto ai controlli nativi, non hanno una finestra propria (sono *WindowLess*) per questo non sono adatti per alcune tecniche di programmazione (come il *subclassing*).

GLOSSARIO

GDI32.DLL
La maggior parte delle funzioni API che permettono di gestire le caratteristiche grafiche del sistema operativo appartengono alla libreria *GDI32.dll (Graphical Device Interface)*. La *GDI32.dll* comunica con le applicazioni Windows-Based per mezzo della *GDI (Windows Graphics Device Interfacce)* e con i drivers della stampante attraverso la *DDI (Device Driver Interfacce)*.

```
   PicFun.PSet (X1, Y1), RGB(r, g, b)
  Next X
 Next fun
End Sub
```

LA FUNZIONE BITBLT

La funzione *BitBlt* è contenuta nella libreria *GDI32 .dll,* cioè la libreria che gestisce le periferiche grafiche. *BitBlt* è l'acronimo di *Bit Block Tansfer,* infatti, la *BitBlt* permette di trasferire un blocco rettangolare di bit da un'immagine ad un'altra in base ad un criterio di trasformazione.
Quindi, questa funzione va bene per il nostro scopo: trasformare un'immagine in base ad una funzione analitica, nella fattispecie trigonometrica.
La sintassi della funzione è la seguente:

```
Declare Function BitBlt Lib "gdi32" (
            ByVal hDestDC As Long,
            _ByVal x As Long,
             ByVal y As Long,
            ByVal nWidth As Long,
            _ByVal nHeight As Long,
            ByVal hSrcDC As Long,
            ByVal xSrc As Long,
            _ByVal ySrc As Long,
              ByVal dwRop As Long) As Long
```

Valore		Descrizione
BLACKNESS	*&H42*	Riempie il rettangolo di destinazione usando il colore associato al valore 0 nella tavolozza fisica (di default il nero).
DSTINVERT	*&H550009*	Inverte i colori del rettangolo di destinazione.
MERGECOPY	*&HC000CA*	Combina i colori del rettangolo di origine, con il pattern modello- specificato, usando l'operatore booleano *AND*.
MERGEPAINT	*&HBB0226*	Combina i colori del rettangolo di origine invertiti (cioè NOT origine) con i colori del rettangolo di destinazione usando l'operatore booleano OR.
NOTSRCCOPY	*&H330008*	Copia il rettangolo di origine invertito, nel rettangolo di destinazione.
NOTSRCERASE	*&H1100A6*	Combina i colori del rettangolo di origine col rettangolo di destinazione usando l'operatore booleano OR ed invertendo i colori risultanti.
SRCAND	*&H8800C6*	Combina i colori del rettangolo sorgente col rettangolo di destinazione utilizzando l'operatore booleano AND.
SRCCOPY	*&HCC0020*	Copia il rettangolo di origine direttamente nel rettangolo di destinazione.
SRCERASE	*&H440328*	Combina i colori invertiti del rettangolo di destinazione (cioè NOT destinazione) con i colori del rettangolo di origine utilizzando l'operatore booleano AND.
SRCINVERT	*&H660046*	Combina i colori del rettangolo di origine col rettangolo di destinazione utilizzando l'operatore XOR.
SRCPAINT	*&HEE0086*	Combina i colori del rettangolo di origine col rettangolo di destinazione utilizzando l'operatore OR.
WHITENESS	*&HFF0062*	Riempie il rettangolo di destinazione usando il colore di destinazione associato al valore 1 nella tavolozza fisica (di default colore Bianco).

Tabella 1: Alcuni valori per dwRop

- **HDestDC -** specifica l'handle delle *PictureBox* che riceve l'immagine;
- **X** e **Y -** sono le coordinate dell'angolo superiore sinistro del rettangolo destinazione;
- **nWidth** e **nHeight -** sono larghezza ed altezza del rettangolo di origine e destinazione;
- **hSrcDC -** è l'handle della PictureBox sorgente;
- **xSrc, ySrc -** sono le coordinata *x* e *y* dell'angolo superiore sinistro del rettangolo di origine;
- **dwRop -** è il codice che specifica l'operazione di scansione dell'immagine. Questo codice definisce come i bit del rettangolo di origine saranno combinati con i bit del rettangolo di destinazione.

Nella **Tabella 1** sono specificati alcuni valori per il parametro *dwRop.*
Ad esempio, per invertire il colore di un'immagine, potete procedere nel seguente modo: su una Form predisponete due PictureBox, nominate *PicSrc* e *PicDes,* e un pulsante. In *PicSrc* inserite un'immagine.

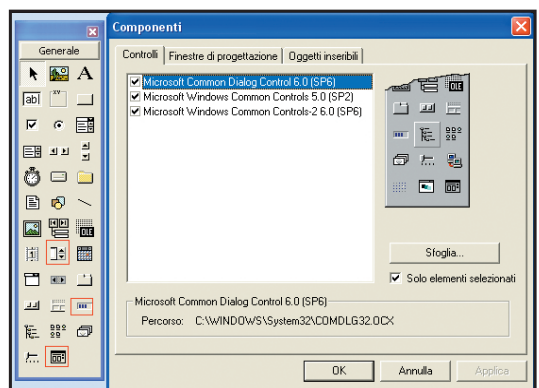



***Fig. 2:** I componenti usati nel Progetto*

Nella procedura del pulsante predisponete le istruzioni che permettono di leggere i pezzi (rettangoli) dell'immagine posta in *PicSrc* e, usando la *BitBlt* con il parametro *dwRop* impostato su *NOTSRCCOPY,* li "trasferiscono" nella *PicDes,* cioè il seguente codice:

```
Private Sub inverti_Click()
 Const NOTSRCCOPY = &H330008
```

```
PicDes.Cls
Hei = PicSrc.ScaleHeight
Wid = PicSrc.ScaleWidth
For X = 0 To Wid
 For Y = 0 To Hei
  DoEvents
  BitBlt PicDes.hDC, X, Y, 1, 1, PicSrc.hDC, _
  X, Y, NOTSRCCOPY
 Next
Next
End Sub
```

I due cicli servono per leggere a pezzi, di forma rettangolare di dimensione *1*, l'immagine sorgente. La dimensione dei rettangoli è data dal valore dei parametri *nWidth* e *nHeight*.

EFFETTI SPECIALI

Su una form prevediamo due serie di comandi che realizzano effetti con diverso grado di difficoltà. I comandi della prima serie permettono di cambiare in modo casuale i colori dell'immagine, deformarne l'aspetto, in base ad una funzione trigonometrica e riprodurla in modo speculare *(Mirror)*. La seconda serie di comandi, più complessa, in base ai valori di alcuni parametri, permette di arrotolare un'immagine, d'ingrandirla o di trascinarla ingrandendola.

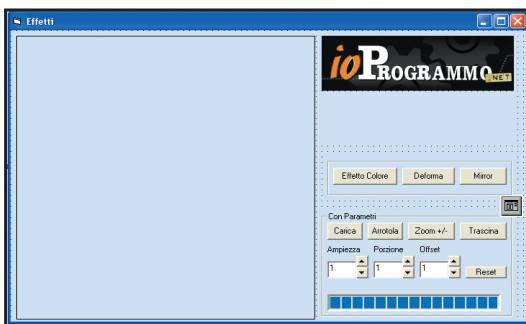



Fig. 3: La Form effetti in fase di progettazione

I parametri che l'utente può impostare sono:

- l'ampiezza dell'immagine trasformata;
- la porzione d'immagine sorgente e trasformata (cioè *nWidth* e *nHeight*);
- la coordinata *Y* del punto da dove inizierà ad essere copiata l'immagine trasformata.

Sulla Form dell'applicazione sono predisposti tre PictureBox: una sorgente, che contiene l'immagine originale, e due destinatarie che conterranno, rispettivamente, l'immagine ricavata con i comandi della prima serie e quella ricavata con i comandi più complessi. Sulla Form, inoltre, sono presenti i controlli che permettono di caricare l'immagine sorgente, di modificare i parametri e di vedere lo stato della trasformazione. Di seguito illustriamo il codice dei comandi principali.

LINE, PSET E

Il metodo Line disegna linee o rettangoli su un oggetto *PictureBox*. La sintassi, basilare, è la seguente:

```
oggetto.Line (x1, y1)
    - (x2, y2), [colore].
```

Dove: *(x1,y1)* indicano le coordinate del punto iniziale e *(x2,y2)* di quello finale della linea; colore, valore *long*, indica il colore RGB utilizzato per disegnare la linea. L'unità di misura utilizzata da Line è specificata da *ScaleMode*. Il Metodo *PSet* imposta un punto in un colore specificato per esempio con la funzione RGB, la sintassi essenziale è la seguente:

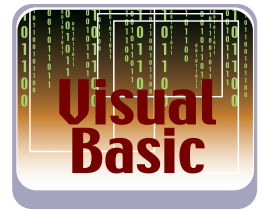
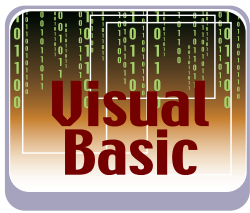
```
oggetto.Pset (x, y),
              [colore].
```

I parametri specificano le coordinate del punto e il suo colore. Infine, la funzione RGB (red,green,blu) in base ai valori dei tre parametri (che rappresentano i tre colori fondamentali: rosso, verde e blu) restituisce un numero che rappresenta il codice di un colore utilizzabile in Visual Basic.

Come al solito il progetto completo lo trovate nel CD allegato alla rivista. In un nuovo progetto bisogna referenziare le seguenti librerie: *Ms Common Dialog Control; Ms Windows Common Controls 5.0* e *Ms Windows Common Controls-2 6.0*. La prima contiene il controllo *CommandDialog* le altre due invece contengono i controlli *UpDown* e *ProgressBar*, utilizzati per modificare i parametri e per visualizzare lo stato della trasformazione.

Fig. 4: Effetto Mirror e Arrotola

Sulla Form sono previste tre PictureBox, nominate *PicDest*, *PicSrc* e *PicMerge*. La *PicSrc* conterrà l'immagine da elaborare, *PicMerge* conterrà l'immagine ricavata con gli effetti di primo livello, mentre *PicDest*, più grande di *PicMerge*, conterrà le immagini ricavate con gli effetti più complessi.
Come accennato i comandi, grazie a dei Fra-

me, sono organizzati in due gruppi. Il gruppo effetti semplici cioè *effetto colore, deforma* e *mirror;* ed il gruppo, nominato con *Parametri*, che comprende i comandi per caricare l'immagine da elaborare, per controllare i parametri e lo stato di avanzamento della trasformazione e per eseguire le seguenti trasformazioni: arrotola, ingrandisci e trascina. Nel Frame con *Parametri* è anche presente il comando *Reset* che consente di impostare ad 1 i valori dei parametri.
Descriviamo la procedura per realizzare gli effetti: *Mirror* e *Arrotola*. Da premettere che le costanti utilizzate e la funzione *BitBlt* vanno dichiarate in un modulo *BAS*. L'effetto *Mirror* lo realizziamo con la funzione lineare *f = picSrc.ScaleWidth - X* applicata alla coordinata *X* della PictureBox.

```
Private Sub Mirror_Click()
 Dim X, Y, f As Double
 Dim Hei, Wid As Integer
 PicMerge.Cls
 Hei = PicSrc.ScaleHeight
 Wid = PicSrc.ScaleWidth
 For X = 0 To Wid
  For Y = 0 To Hei
   f = Wid - X
   DoEvents
   BitBlt PicMerge.hDC, f, Y, _
   1, 1, PicSrc.hDC, X, Y, SRCCOPY
  Next
 Next
End Sub
```

SCALE

Le proprietà *ScaleMode, ScaleHeight* e *ScaleWidth* permettono di specificare il tipo e la dimensione del sistema di coordinate di un oggetto:

- ***ScaleHeight* imposta, o fornisce, il numero di unità per l'asse verticale (*ScaleWidth* per quello orizzontale), per esempio *ScaleHeight=50* imposta a 50 le unità massime per l'asse verticale.**
- ***Scale Mode*, invece, imposta o restituisce il tipo di unità di misura utilizzata per un oggetto (i valori sono *0=definita dall'utente, 1= Twip ..., 7=centimetri ...*).**

Come nelle procedure precedenti i due cicli *For* servono a leggere i vari rettangoli di bit che formano l'immagine. I rettangoli sono letti alla posizione *[X,Y]* e scritti alla posizione *[wid – X,Y]*, quindi si deduce che se *Wid =10* un rettangolo di bit letto alla posizione *[3,1]* è copiato nella posizione *[7,1]*. Descriviamo come realizzare l'effetto *Arrotola*, cioè la procedura che permette di trasformare un'immagine piatta in un'immagine cilindrica. A tal fine vengono utilizzate combinazioni delle funzioni trigonometriche *Sin* e *Cos;* però lasciamo a voi, se ne avete voglia, l'onere di capire come possano due funzioni trigonometriche produrre un cerchio e quindi un cilindro.

```
Private Sub arrotola_Click(
 Dim X, Y, A, B, PiG As Double
 Dim Hei, Wid, amp, s, offset As Integer
 PicDest.AutoRedraw = True
 PicSrc.AutoRedraw = True
 PicDest.Cls
 Hei = PicSrc.ScaleHeight
 Wid = PicSrc.ScaleWidth
 PiG = (6.28 / 360)
 ProgressBar1.Max = Wid + 1
 ProgressBar1.Value = 0
 If txtampiezza = 1 And txtoffset = 1 _
 And txtspessore = 1 Then
  s = 4
  offset = 100
  amp = 70
  txtspessore = s
  txtoffset = offset
  txtampiezza = amp
 Else
  s = txtspessore
  offset = txtoffset
  amp = txtampiezza
 End If
 For X = 0 To Wid
  DoEvents
  ProgressBar1.Value = ProgressBar1.Value + 1
  A = amp * Sin(PiG * X)
  For Y = 0 To Hei
   B = amp * Cos(PiG * X)
   DoEvents
   BitBlt PicDest.hDC, (PicDest.ScaleWidth / 2) + A _
   , Y + B + offset, s, s, PicSrc.hDC, X, Y, SRCCOPY
  Next
  PicDest.Refresh
 Next
End Sub
```

Nella procedura facciamo notare, al di là della trigonometria, che quando i valori dei TextBox *(parametri)* sono tutti uguali a *1* le variabili *s (spessore), offset* e *amp* (ampiezza o modulo della funzione) sono impostati su dei valori di default che permettono di visualizzare l'effetto. Inoltre notate che nella procedura è gestito il valore corrente della *ProgressBar1* (questo è incrementato nel ciclo più esterno) e che l'immagine trasformata viene creata intorno al punto di coordinate *[picDest.ScaleWidth / 2, offset]*.
Altri due effetti interessanti sono: *inversione*

colore in modo casuale (basato su *MERGE-PAINT*) e *Zoom* +/- (ingrandimento e rimpicciolimento).
Di seguito presentiamo le procedure.

```
Private Sub merge_Click()
 Dim X, Y As Double
 Dim Hei, Wid, I As Integer
 I = Int((15 * Rnd))
 'tra zero e 15
 PicMerge.BackColor = QBColor(I)
 Hei = PicMerge.ScaleHeight
 Wid = PicMerge.ScaleWidth
 DoEvents
 For X = 0 To Wid
  For Y = 0 To Hei
   DoEvents
   BitBlt PicMerge.hDC, X, Y, 2, 2, _
   PicSrc.hDC, X, Y, MERGEPAINT
  Next
 Next
End Sub
```

Nella procedura precedente il colore della *PicMerge* è generato con *QBColor*, in base ad un numero casuale compreso tra *0* e *15 (0=nero, … 15= Bianco brillante)*

```
Private Sub Zoom_Click()
 Dim X, Y, A, B As Double
 Dim Hei, Wid As Integer
 PicDest.AutoRedraw = True
 PicSrc.AutoRedraw = True
 PicDest.Cls
 Hei = PicSrc.ScaleHeight
 Wid = PicSrc.ScaleWidth
 ProgressBar1.Max = Wid + 1
 ProgressBar1.Value = 0
 offset = txtoffset
 amp = txtampiezza

 If txtampiezza = 1 And txtoffset = 1 _
  And txtspessore = 1 Then
  s = 3
  offset = 1
  amp = 3
  txtspessore = s
  txtoffset = offset
  txtampiezza = amp
 Else
  s = txtspessore
  offset = txtoffset
  amp = txtampiezza
 End If

 For X = 0 To Wid
  DoEvents
  ProgressBar1.Value = ProgressBar1.Value + 1
   If amp < 0 Then
    amp = 1 / Abs(amp)
   End If

   A = amp * X
   For Y = 0 To Hei
    B = amp * Y
    DoEvents
    BitBlt PicDest.hDC, A, B + offset, s, s, _
    PicSrc.hDC, X, Y, SRCCOPY
   Next
   PicDest.Refresh
  Next
 End Sub
```

Notate che il rimpicciolimento delle immagini si ottiene quando l'ampiezza *(txtampiezza)* è minore di *-1*.

AUTOREDRAW

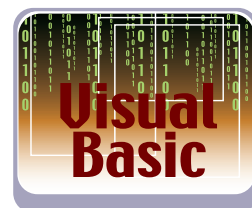
La proprietà *AutoRedraw* è un Booleano che attiva o disattiva l'aggiornamento automatico di una PictureBox. In particolare se è impostata su *False* l'immagine è mostrata solo sullo schermo e non è memorizzata in un'immagine residente in memoria. Se si utilizza la funzione *BitBlt* su una PictureBox con *AutoRedraw* impostata su *True* per vedere l'immagine, trasformata, bisogna per forza richiamare il metodo *Refresh;* mentre se la proprietà è impostata su *False* non è necessario richiamare il metodo *Refresh* ma se la *Form* perde il *focus* l'immagine viene cancellata (o rovinata), dato che non è archiviata in memoria.

Infine presentiamo il codice da predisporre nel controllo *UpDown* che permette di modificare il parametro ampiezza, cioè il valore contenuto in *TxtAmpiezza*.

```
Private Sub UpDown1_DownClick()
 txtampiezza = txtampiezza - 1
End Sub

Private Sub UpDown1_UpClick()
 txtampiezza = txtampiezza + 1
End Sub
```

CONCLUSIONI

Nel corso dell'articolo oltre a descrivere l'utilizzo della funzione *BitBlt*, abbiamo visto come elaborare e tracciare le funzioni matematiche e come legarle alla trasformazione di un'immagine.
In conclusione vi invitiamo a creare qualche simpatico effetto, magari usando le funzioni trigonometriche derivate.

Massimo Autiero

Programmare con Darwin

Se ne parla ormai da moltissimo tempo, ma cosa sono e come funzionano gli algoritimi genetici? Come possiamo usarli, in modo intelligente, per sviluppare software che migliora nel tempo?

Conoscenze richieste
J2SE, JGAP

Software
Java 2 Standard Edition SDK 1.4 o superiore

Impegno

Tempo di realizzazione

Pur non essendo dei biologi proveremo, in questo articolo, a parlare dell'evoluzione delle specie. Cercheremo di implementare un algoritmo che simuli tale evoluzione. Cerchiamo di spiegarci meglio. Si parte dal concetto che l'essere umano si è evoluto nel tempo "adattandosi" all'ambiente circostante. In Africa la pelle scura è necessaria per difendersi dal sole cocente, in Europa questo tratto genetico non si è sviluppato perché l'uomo europeo non aveva bisogno di adattarsi a questo tipo di problema. In sostanza, il meccanismo evolutivo, tenta di trovare una soluzione ai problemi che l'ambiente esterno gli sottopone. Nel tempo, abbiamo scoperto che scientificamente questo tipo di evoluzione è dovuta al carattere ereditario del DNA e alla sua capacità di ricombinare i cromosomi per creare un individuo nuovo e geneticamente "migliore".
E se adottassimo lo stesso schema per la risoluzione dei nostri problemi informatici? Supponete di volere creare una soluzione per il gioco del master mind. Potremmo partire da un certo numero di soluzioni, per poi ricombinarle sulla base di un grado di "soddisfazione" del problema... quello che abbiamo così rozzamente enunciato è un modo di introdurre gli "Algoritmi Genetici".

DA DARWIN ALLA SCOPERTA DEL DNA

Le due cose che dobbiamo tenere ben presente quando affrontiamo gli algoritmi genetici sono: aver chiare le leggi fondamentali che hanno permesso l'evoluzione della vita sulla terra e dimenticarci di come siamo solitamente abituati a programmare, ma lasciare che le soluzioni del nostro problema "evolvano naturalmente".
Sono passati ormai quasi 150 anni da quando Darwin enunciò nel suo libro "L'origine della specie", una teoria destinata a rivoluzionare la concezione occidentale della vita.
È ben noto che, uno dei principi cardine di tale teoria è dato dalla "selezione naturale"; volendo dare una definizione più formale di tale principio si può affermare che: secondo la teoria evoluzionistica di Darwin l'ambiente subendo mutamenti, opera una selezione naturale graduale sulla grande variabilità che ogni carattere presenta nelle singole specie, scegliendo così le forme di volta in volta più adatte. Naturalmente, all'epoca, Darwin aveva formulato una teoria sulle osservazioni che egli stesso aveva effettuato durante i suoi viaggi, ma non spiegava il meccanismo grazie al quale l'evoluzione permetteva il passaggio delle caratteristiche da una generazione all'altra, né come fosse possibile che, nel lungo termine, la teoria evolutiva desse origine a specie diverse, pur discendendo dalla stessa base, ad esempio perché esistono contemporaneamente l'uomo e la scimmia?
Le critiche alle teoria di Darwin vennero stroncate, dalla scoperta del DNA da parte di Watson & Circk. Dopo tale scoperta si è infatti compreso come i caratteri ereditari si trasmettano da una generazione alla successiva e come siano inoltre possibili mutazioni tra generazioni lontane nel tempo (si pensi che l'uomo e lo scimpanzé hanno in comune il 98% del patrimonio genetico).
Quello che ci accingiamo a fare e proprio generare una "popolazione di soluzioni" formata da individui che sono identificati da un *"DNA digitale"* . Porremo tale popolazione in un ambiente che sarà rappresentato proprio

dal problema che vogliamo risolvere. A questo punto non dovremmo far altro che aspettare l'evoluzione della specie, finché il meccanismo di ricombinazione cromosomica non generi una soluzione per noi accettabile.
È naturale che, per realizzare un simile sistema dobbiamo innanzitutto conoscere i meccanismi che sono alla base del funzionamento del DNA.
Il DNA è un acido nucleico che contiene le informazioni genetiche, sotto forma di sequenza di basi azotate.

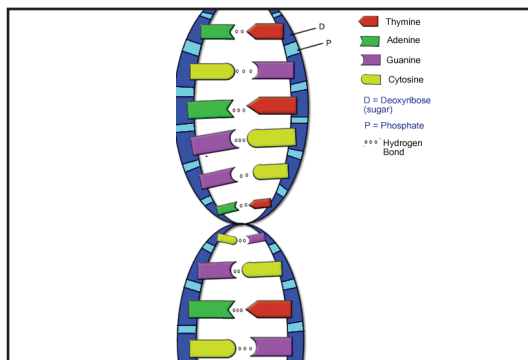



Fig.1: *Struttura schematica del DNA*

Esso è contenuto nel nucleo di ogni cellula ed è "organizzato" in cromosomi (46 per il genere umano), che lunghi filamenti di DNA "arricciati" su se stessi.
Nel nostro modello informatico, semplificheremo questo aspetto in modo tale che, ogni individuo sia identificato da un unico cromosoma, e a variare sarà la lunghezza del cromosoma stesso a seconda di quante informazioni dovrà codificare.
A loro volta i cromosomi possono essere divisi in "unità fondamentali", dette geni.

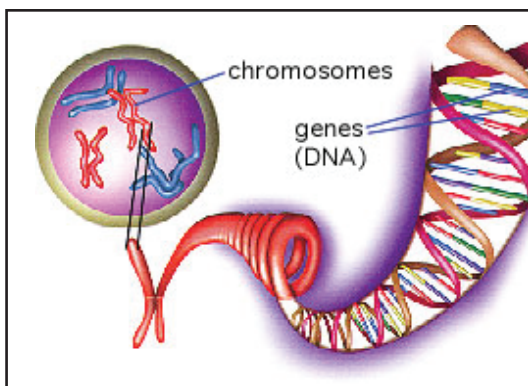



Fig.2: *Relazione tra gene e cromosoma*

Benché in biologia le cose siano più complesse di quanto stiamo dicendo, ci accontenteremo di dire che ogni gene è portatore di una specifica caratteristica di un individuo.
Per essere ancor più precisi, chiameremo la manifestazione di un gene "allele". Facciamo un esempio, puramente teorico ma che può aiutarci a capire quanto detto fin qui: supponiamo di avere un unico cromosoma, che identifichi una rosa composto da *N* geni, Ogni gene codificherà una specifica caratteristica del fiore stesso (lunghezza dello stelo, numero di spine, larghezza dei petali etc..), e supponiamo che il gene numero 3 sia il responsabile del colore dei petali.
A questo punto, se una rosa mostra dei petali rossi diremo che il terzo gene è un allele rosso. Per fare un'analogia con un mondo più vicino al nostro potremmo dire che un gene rappresenta il tipo di dato (int, double, char, etc...) mentre un allele è l'istanza del dato stesso (5, 0.3, 'a', etc...).

MODELLO INFORMATICO DEL CROMOSOMA

Ricordando tutto quanto detto fin ora e tenendo presente che il linguaggio che utilizzeremo è Java, cioè un linguaggio Object Oriented, il nostro cromosoma può essere ben modellizzabile da una stringa di oggetti.

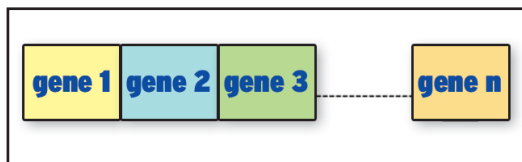



Fig.3: *Struttura informatica di un gene*

Ogni oggetto costituente sarà quindi identificato con "gene".
Nelle librerie che utilizzeremo, l'allele è rappresentato come un Object generico, recuperabile grazie al metodo getAllele, anche se questo risulterà più chiaro quando metteremo mano al codice; per ora ci basta aver chiaro il progetto d'implementazione generale.
Nella libreria JGAP il gene è un'interfaccia che ha varie classi che la implementano, ad esempio se volessimo creare un gene che abbia un allele di tipo double basterà:

```
Gene gene = new DoubleGene(0,30);
```

I due argomenti del costruttore, rappresentano il range di valori che il gene può assumere. Se invece vogliamo creare un cromosoma, dobbiamo innanzitutto definire un array di geni e poi passarlo come argomento al costruttore della classe *Chromosome:*

```
Gene[] genes = new Gene[]{new DoubleGene(
```

```
                    0,100), new DoubleGene(0,100)};
Chromosome sampleChromosome =
                          new Chromosome(genes);
```

Fin qui abbiamo descritto un cromosoma preso singolarmente, ora però vediamo come funziona la parte dinamica degli algoritmi genetici, ossia come essi possano incrociarsi dando vita alla generazione successiva e come possano anche mutare.

CROSSOVER

Il meccanismo di crossover è quello che permette di generare un nuovo cromosoma a partire da altri due. Il procedimento è ben illustrato in **Figura 4**.

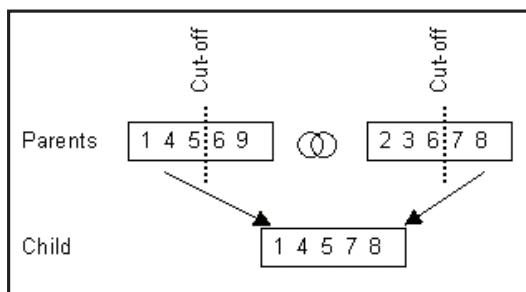



Fig.4: ***Meccanismo della generazione di un gene figlio da due genitori***

Si stabilisce un punto di cut-off, ossia il punto in cui ogni cromosoma si divide in due parti convenzionalmente nominate *Testa* e *Coda*. Come è ben illustrato in figura, la generazione di un nuovo cromosoma è data dall'incrocio della testa di uno e la coda dell'altro.
Naturalmente, la scelta di dove collocare tale punto, dipende soprattutto dalla codifica del cromosoma stesso, ossia da ciò che ogni gene rappresenta. In realtà, per cromosomi "considerevolmente lunghi", che implicano la ricerca su uno spazio di soluzioni ad alte dimensioni, si usano punti di crossover multipli; per ora, non complichiamoci la vita e dividiamo semplicemente ogni cromosoma in due parti.

MUTAZIONI

Possiamo già intuire che il crossover "rimescola" i caratteri, tuttavia, dopo un po' di combinazioni, ritroveremo gli stessi individui; è un po' come quando un ristretto nucleo di individui si accoppiano sempre tra loro, difficilmente ci sarà un individuo con caratteristiche considerevolmente diverse dagli altri. Per superare questo limite si addotta la mutazione, cioè scegliere secondo un criterio arbitrario un gene e si cambia il valore dell'allele relativo. Queste due procedure sono anche chiamate "operatori genetici", la cosa interessante da capire è come far agire questi operatori. Infatti all'inizio dell'articolo, non a caso, avevamo accennato alle interazioni tra ambiente e caratteri genetici degli individui, sintetizzando il tutto sotto il principio di selezione naturale.

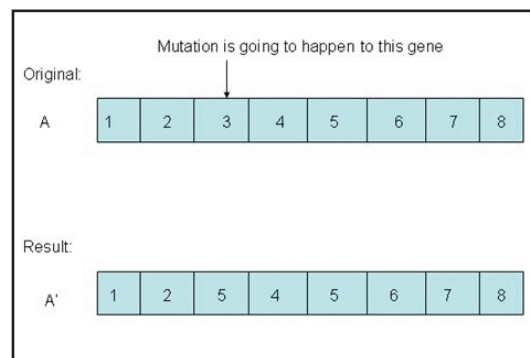



Fig.5: ***Struttura dei buffer dopo il processo di allocazione***

Quindi, l'ultimo passo che ci rimane da compiere è definire un criterio per valutare quanto ogni cromosoma (o individuo) sia adatto ad un ambiente.

FITNESS FUNCTION

Questo è forse l'aspetto cruciale di tutti i nostri discorsi. Tradotta letteralmente, *fitness function* significa "funzione di benessere"; l'idea è quella di associare un numero che quantifichi il grado di adattamento di un individuo ad un ambiente.
Abbandonando il contesto biologico e venendo al nostro, possiamo dire che la fitness function stimerà la bontà della soluzione ad un determinato problema.
Per capirci meglio facciamo l'esempio più banale che possa venirci in mente: definiamo un cromosoma composto da soli 0 o 1 e diciamo che più 1 un cromosoma contiene e migliore sarà la nostra soluzione (è naturale che il miglior cromosoma sarà quello contenente tutti 1, ma a noi adesso non interessa).
Possiamo quindi definire una fitness function che conti quanti 1 sono presenti in un cromosoma.

```
F(00100)=1
F(10110)=3
```

e così via...

IMPLEMENTAZIONE DI UN ALGORITMO GENETICO

1. Inizializza una popolazione formata da *N* cromosomi ciascuno dei quali composto da *K* geni.
2. Calcolare l'idoneità di ogni cromosoma *x* mediante la *fitness function f(x)*.
3. Ripetere i seguenti passaggi finché non si è raggiunta una popolazione di *N* cromosomi:

   a) Seleziona una coppia di cromosomi.
   b) Genera una nuova coppia cromosomica discendente mediante il crossover con una probabilità Pc
   c) Se l'incrocio non avviene clonare la coppia di cromosomi così come è.
   d) Muta ogni gene dei due cromosomi così ottenuti con una probabilità *Pm*.
4. Rimpiazza la generazione precedente con quella nuova ottenuta dai passi **1-3.**
5. Tornare al punto **2** finché non viene verificata la condizione di *STOP.*

Chiariamo un paio di punti fondamentali.

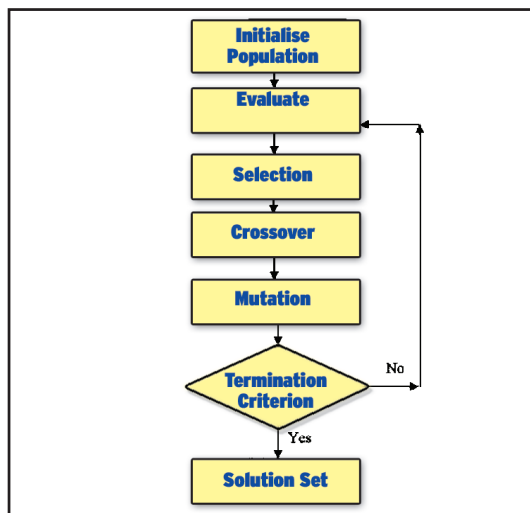



Fig. 4: Flow-chart dell'algoritmo

La seleziona descritta nel punto **3a** avviene in rapporto al valore di *fitness;* in particolare maggiore sarà tale valore maggiore sarà la probabilità del cromosoma ad essere selezionato per dar vita ai cromosomi della generazione successiva. Questo, rispetta il principio che gli individui "più forti" hanno una maggiore possibilità di accoppiarsi e quindi di trasferire i propri caratteri ereditari alla generazione successiva. Quando poi si parla di probabilità di mutazione, si intende che se fissiamo ad esempio *Pm =0,05,* ciò implica che ogni cromosoma generato nel punto **3c** avrà una probabilità di vedere uno dei suoi geni mutato pari al 5%. Analoga considerazione, può essere fatta per la probabilità di ricombinazione del punto **3b**. Per motivi di brevità e poiché questo problema esula dalle finalità di questo articolo non ci soffermeremo su come simulare un fenomeno aleatorio (o probabilistico) in un calcolatore, ma sottolineo comunque l'importanza di tale aspetto.

TROVIAMO UNA SOLUZIONE PER MASTERMIND

Per chi di voi non conoscesse questo gioco ricapitoliamo le regole nella versione più "hard". Ci sono 5 posizioni che vanno riempite con altrettante palline. I colori disponibili per le palline sono 5 e non ci sono limiti sulle ripetizioni (ad esempio si può scegliere di prendere tutte 5 le palline di un unico colore).
Un giocatore, che chiameremo banco, sceglie quindi una successione di palline in maniera del tutto arbitraria e la tiene nascosta all'altro giocatore. Di contro, quest'ultimo deve indovinare la combinazione scelta dall'avversario, sottoponendogli di volta in volta una sua combinazione alla quale il banco dovrà rispondere, dicendo quante palline hanno lo stesso colore e la stessa posizione. Ad esempio, se il banco sceglie ROSSO-GIALLO-BIANCO-ROSSO-BLU alla combinazione proposta dal giocatore VERDE-NERO-BLU-ROSSO-BLU esso risponderà con 1, perché l'unica

IL CODICE DEL DNA

In realtà il DNA può essere considerato un codice ridondante, in quanto le sue quattro basi adenina-timina e citosina-guanina, si possono accoppiare solo secondo questo schema; in altre parole, se su un filamento è presente la base A, nella corrispondente posizione dell'altro filamento dovrà trovarsi una T e mai una C o una G. *(Figura 1)*

SERIALIZZAZIONE

In alcuni casi la generazione di una popolazione di soluzioni può impegnare notevoli risorse in termini di tempo, in altri casi sarebbe opportuno distribuire l'evoluzione cromosomica a più computer per ottenere risultati diversi e in modo più veloce.
JGAP contiene un meccanismo che consente di esportare le soluzioni in formato XML, l'esempio riportato anche sul sito di JGAP http://jgap.sourceforge.net/doc/marshalling.html **è abbastanza semplice.**
Per l'esecuzione di questo esempio è necessario avere installato il parser *Xerces*

```
public static saveStrategyState(
Genotype a_currentPopulation, string
a_saveToFilename )
throws Exception
{
  // Converte un Genotype in un DOM
object.
  Document xmlRepresentation =
    XMLManager.representGenotype
    AsDocument(a_currentPopulation);
  // Scrive il documento DOM sul
    disco in formato XML
  Writer documentWriter = new
    FileWriter( a_saveToFilename );
......
```

è necessario poi implementare i vari serializer, sul sito di JGAP viene riportato l'esempio completo, anche per la lettura del documento XML e la riconversione in genotipo.

pallina coincidente è quella nella quarta posizione. Quando un giocatore ha indovinato la combinazione i ruoli si invertono; alla fine vince chi è riuscito ad indovinare le mosse dell'avversario nel numero minore di tentavi.

MANO AL CODICE

La prima cosa da fare è pensare a che tipo di cromosoma utilizzare e definire la fitness function. Le due cose non sono così slegate una dall'altra, infatti affrontiamo entrambi gli aspetti insieme. La libreria JGAP prevede una classe astratta *FitnessFunction,* in cui bisogna implementare il metodo

```
protected double evaluate(Chromosome arg0)

Estendiamo quindi tale classe con la nostra MMFitness

public class MMFitness extends FitnessFunction {

    private int[] targetSeq;
    public MMFitness(int[] sol){
        targetSeq = new int[sol.length];
        System.arraycopy(sol,0,targetSeq,0,sol.length);
}
```

Come possiamo notare, abbiamo definito un costruttore in cui il banco "nasconde" la propria soluzione sotto forma di array di interi, abbiamo infatti sostituito i colori con dei valori numerici, ma il gioco rimane lo stesso.
Ora dobbiamo implementare il metodo evaluate, per far ciò dobbiamo anche definire la classe *Chromosome*.

```
...
protected double evaluate(Chromosome arg0) {
    double fit = 0;
    if(arg0.size() == targetSeq.length){
        for(int i=0; i<targetSeq.length; i++)
        if(targetSeq[i]== ((Integer)arg0.getGene(i)
                                        .getAllele()).intValue())
        fit++;
    } else
    fit = Double.NaN;
    return fit;
}
```

Naturalmente, la prima condizione da dover essere rispettata è che il cromosoma abbia la medesima lunghezza della sequenza del banco, dopo di che, si aumenta una unità il punteggio di fitness per ogni valore inserito nella corretta posizione.
Vediamo ora come far girare il tutto:

Giochiamo con 5 palline e 5 colori, quindi abbiamo soluzioni, una popolazione di 900 individui dovrebbe essere più che sufficiente. Procediamo allora come segue:

```
    ...
int[] target = new int[]{5,2,4,4,3};
FitnessFunction function = new MMFitness(target);
```

Dobbiamo poi definire tutti quei parametri illustrati al passo **3**, fortunatamente JGAP ci mette già a disposizione una serie di configurazioni che, per il nostro scopo, vanno benissimo, quindi semplicemente creiamo un cromosoma prototipo e passiamolo alla configurazione scelta, sarà lei ad occuparsi del resto.

```
Gene[] genes = new Gene[5];
Arrays.fill(genes,new IntegerGene(1,5));
Chromosome sample = new Chromosome(genes);
Configuration conf = new DefaultConfiguration();
Genotype population = null;
try {
    conf.setFitnessFunction(function);
    conf.setSampleChromosome(sample);
    conf.setPopulationSize(900);
    population = Genotype.randomInitialGenotype(
                                                conf);
    ...
```

A questo punto, non ci rimane che far evolvere la popolazione così inizializzata fino a che non troviamo la soluzione.

```
do
{
    population.evolve();
    Chromosome best =
                population.getFittestChromosome();
    System.out.println("Migliore soluzione:" + print(
                                                best));
    fit = best.getFitnessValue();
}
while(fit < 5);
```

CONCLUSIONI

In questo articolo, abbiamo presentato i concetti che sono alla base degli algoritmi genetici, per poi fare un giochino che prevedeva un gran numero di possibili combinazioni.
In realtà, gli algoritmi genetici hanno applicazioni in un gran numero di campi, soprattutto per quei problemi di difficile soluzione come possono essere gli NP.

Andrea Galeazzi

L'ORIGINE DELLE TEORIE DARWINIANE

Darwin formulò le sue teorie traendo spunto anche da "Principi di Geologia" di Charles Lyell, secondo il quale i mutamenti geologici erano in realtà sempre in atto, ma anche così lenti da essere appena percettibili nell'arco di una vita umana. Analogamente, Darwin ritenne che l'evoluzione delle specie era in continuo divenire ma che la storia dell'umanità sia troppo breve per accorgersene. In effetti anche noi ci renderemo conto di come tali algoritmi abbiamo bisogno di varie generazioni di soluzioni prima di generarne una ottima o sub-ottima.

Laureato all'università Politecnica delle Marche, lavora presso un gruppo nazionale leader nel campo delle tecnologie e dei servizi IT. Nei limiti della disponibilità di tempo risponde all'indirizzo andreagaleazzi@fsfe.org

Leggere e scrivere su una connessione seriale sotto Windows

L'idea di trattare questo argomento nella rubrica Express nasce dalla richiesta fatta da un lettore sul sito di ioProgrammo. Sotto ambiente Windows leggere e scrivere sulle porte seriali è molto semplice. Inoltre l'argomento è trattato bene nella documentazione in linea sul sito di microsoft msdn alla sezione *"Serial Communications in Win32"*. Noi vedremo un semplice esempio, per prendere familiaritàre con l'argomento in questione. In particolare ci occuperemo della comunicazione con il modem,attraverso la porta COM, al fine di ottenere alcune informazioni riguardo l'apparecchio. La tecnica presentata in questo articolo, nella maggior parte dei casi, funziona anche nel caso di comunicazione con le porte COM virtuali associate alle porte USB ma in alcuni casi ciò non è possibile quindi dobbiamo far riferimento alla documentazione fornita con il particolare dispositivo da controllare o con il quale comunicare. Ciò di cui abbiamo bisogno, per realizzare il seguente progetto, è un modem seriale o USB un compilatore C++, un editor di testo e un po' di dimestichezza con il linguaggio C++. La conoscenza di qualche comando AT potrebbe tornare utile.

Stefano Vena

‹1› SI PARTE!

```
#include <cstdlib>
#include <cstdio>
#include <windows.h>
using namespace std;
int main(int argc, char *argv[])
{ HANDLE hCom;
  DCB dcb;
  char comm[10]= "\0";
  sprintf(comm,"\\\\.\\COM%d", 1);
```

La prima cosa da fare è creare un nuovo file e chiamarlo, ad esempio, *"seriale.cpp"* poi, inserite le direttive per l'inclusione dei file di intestazione necessari, andiamo a scrivere il codice del main. Le variabili necessarie alla connessione sulla porta COM sono due: *hCom* e *dcb*. La prima memorizza il valore handle associato alla connessione la seconda è una struttura e viene utilizzata per settare alcuni parametri della porta seriale. A questo punto scegliamo quale porta aprire e inseriamo il nome in una terza variabile dichiarata come stringa.

‹2› LA CONNESSIONE

```
hCom = CreateFile(comm,
            GENERIC_READ | GENERIC_WRITE, 0, NULL,
                            OPEN_EXISTING, 0, NULL );
if( hCom == INVALID_HANDLE_VALUE)
{ printf("Errore nella connessione\n");
  return -1;
}
```

Ebbene si! Il Windows gestisce le porte seriali come se fossero dei file. Quindi andiamo a creare un nuovo file facendo attenzione ai valori passati alla funzione *CreateFile* il primo identifica il percorso della porta. La sintassi per identificare la porta l'abbiamo appresa nel passo precedente. Il secondo parametro identifica il modo di lettura e scrittura. Il terzo parametro imposta l'accesso alla risorsa in modo esclusivo, ovvero la porta appena aperta non può essere condivisa con il resto del sistema. Il quarto parametro contiene le informazioni su eventuali impostazioni di sicurezza, nel nostro caso non necessarie. Gli altri parametri sono scelte forzate per le periferiche di comunicazione eccezione fatta per il penultimo parametro che identifica la funzione di overlap. *Overlap* (valore 1) è sinonimo di accesso asincrono (un accesso sincrono avviene scegliendo il valore 0).

‹3› SETTAGGI

```
GetCommState(hCom, &dcb);
 //Modifichiamo i settaggi della porta
  dcb.BaudRate = 9600;
  dcb.ByteSize = 8;
  dcb.Parity = NOPARITY;
  dcb.StopBits = ONESTOPBIT;
SetCommState(hCom, &dcb);
```

In questo passo andiamo ad ottenete i settaggi correnti della porta attraverso l'utilizzo della funzione *GetCommState*, la quale riempe la struttura dcb con i valori attuali.
Poi andiamo a modificare i parametri che ci interessano e aggiorniamo lo stato della porta attraverso la funzione *SetCommState*

‹4› PREPARAZIONE DEL COMANDO

```
  DWORD dwBytesWritten = 0;
  DWORD dwBytesRead = 0;
  DWORD dwCount = 255;
  char Comando [dwCount];
  char result [255];
  strcpy( result , "" );
  //Preparazione Comando
  strcpy( Comando , "ATI3\n" );
```

La comunicazione con la porta, prevede l'utilizzo di un buffer di caratteri che inizializzeremo con una dimensione forfettaria di 255 celle. È necessario un altro buffer per la ricezione dei comandi di risposta. Andiamo, allora, a creare un secondo vettore di caratteri e lo inizializziamo con una stringa vuota. Ora siamo pronti per preparare il comando da inviare il modem.

‹5› INVIO DEL COMANDO

```
//Invio del Comando
if( WriteFile(hCom, Comando strlen(Comando),
                          &dwBytesWritten, NULL) )
{
```

Inviare il comando appena creato è molto semplice. Questa operazione viene svolta dalla funzione *WriteFile*. I parametri della funzione sono, in ordine, il valore *Handle* della porta, il comando, il numero di byte effettivi del comando, il puntatore alla variabile che riceve il numero di byte scritti, e il puntatore alla struttura che contiene le informazioni riguardanti la scrittura in modalità *overlap*.

‹6› RICEVERE LA RISPOSTA

```
while( strstr(Comando, "OK") == NULL )
{
  ReadFile(hCom, Comando, dwCount,
                             &dwBytesRead, NULL);
  strncat( result, Comando, dwBytesRead );
 }
}
 printf( result );
 CloseHandle(hCom);
 system("PAUSE");
 return 1;
}
```

Il modem risponde alla richiesta con un ordine preciso. Prima restituisce il comando richiesto, poi invia il valore richiesto poi invia la stringa *"OK"* se l'operazione è stata compiuta regolarmente oppure risponde con *"ERROR"* se la richiesta non è andata a buon fine. Ipotizziamo l'assenza di errori,quindi finché non otteniamo il valore *OK* concateniamo le stringhe ottenute alla stringa *result*, una volta ottenuto l'ok dal modem usciamo dal ciclo e stampiamo il risultato.

Realizzare una piccola classe con le funzioni di un cronometro

In questo Express andremo a vedere come realizzare un semplice cronometro. Ad esempio potremmo farne uso per calcolare la velocità di rendering di una scena oppure per calcolare la velocità di trasferimento di un download da internet oppure potremmo fare delle stime, all'interno di una procedura molto lunga, del tempo necessario allo svolgimento totale dell'operazione in corso. Come vedremo tra qualche riga, la classe è molto semplice da realizzare e non richiede grosse conoscenze. Per ottenere gli intervalli di tempo utilizzeremo il metodo *clock()* che viene dichiarato nell'header *"time.h"* del C++ quindi ci garantisce la piena portabilità del codice. Il metodo *clock()* restituisce il tempo del processore a partire dall'avvio del programma. Per ottenere il tempo (approssimativo) in secondi dobbiamo effettuare una divisione per la costante *CLOCKS_PER_SEC*. Ciò di cui abbiamo bisogno come al solito è il nostro compilatore C++ preferito, un editor di testo e un po' di dimestichezza con il linguaggio C++.

Stefano Vena

‹1› LE PRIME RIGHE DI CODICE

```
#ifndef STOPWATCH_H
#define STOPWATCH_H
#include <time.h>
class Stopwatch
{   private:
      clock_t start, finish, step;
      bool run;;
```

La prima cosa da fare è creare un nuovo file e chiamarlo *"Stopwatch.h"*. Dopo aver inserito i comandi del preprocessore ed aver incluso il file *>time.h>* andiamo a definire la classe e i suoi membri. Per la realizzazione del cronometro sono necessarie quattro variabili. Le tre variabili di tipo *clock_t* vengono utilizzate per memorizzare il tempo di partenza, di fine e l'intervallo di tempo trascorso tra l'avvio e l'arresto del cronometro. L'unica variabile booleana dichiarata funge da flag e indica se il cronometro è in esecuzione o meno. Dal momento che il codice di ogni metodo che andremo a scrivere è minimo possiamo scrivere l'intera classe inline, ovvero criveremo tutto il codice nella definizione della classe..

‹2› AVVIO

```
public :
 Stopwatch()
 {  run = false;
    step = (clock_t) 0;
 }
 Stopwatch(bool start)
 {  step = (clock_t) 0;
    if( start ) Start();
      else run = false;
 }
```

La classe prevede due costruttori: il primo è quello di default e assegna il valore *false* al flag *run*. Il secondo prende come parametro un booleano che indica se il cronometro deve essere avviato subito dopo la sua creazione. Se il cronometro deve partire allora viene chiamata la funzione *Start()*, che vedremo fra poco, altrimenti settiamo il flag su *false*. Entrambi i costruttori inizializzano la variabile *step* a 0.

‹3› START

```
clock_t Start()
{
  if( run ) Stop();
  run = true;
  start = clock();
  return step;
}
```

Andiamo a vedere come avviare il nostro cronometro. Se il cronometro è già in esecuzione allora lo arrestiamo. Dopo aver gestito questa eventualità andiamo ad avviare nuovamente il conteggio del tempo. Impostiamo a true il flag che indica lo stato di esecuzione e memorizziamo il tempo del processore attuale. Ora possiamo ritornare il valore dell'intervallo di tempo trascorso (se ne è trascorso) e passare alla prossima funzione.

‹4› STOP

```
clock_t Stop()
{ if( run )
  {  finish = clock();
      step = (finish - start);
     run = false;
     return step;
  }
  else
     return 0;
}
```

.Se siamo in esecuzione allora è necessario ottenere il tempo attuale del processore e sottrarlo a quello registrato all'avvio del cronometro quindi andiamo a memorizzare questo valore nella variabile step;
Poi è necessario aggiornare il flag *run* a *false* e ritornare l'intervallo trascorso. Se il cronometro è fermo allora ritorniamo il valore zero.Questa funzione è l'ultima funzione di base del cronometro.

‹5› ACCESSORI

```
clock_t ClocksCount() const
{ if( run ) return ( clock() - start);
  else return step; }
static double ToSeconds(const Stopwatch& sw )
{ return  ToSeconds( sw.ClocksCount() ); }
static double ToSeconds( clock_t time )
{ return  ((double) time )/CLOCKS_PER_SEC; } };
#endif
```

La prima funzione, ovvero *ClocksCount()* ci restituisce, se disponibile, il tempo parziale senza interrompere il conteggio complessivo. Le altre due funzioni, dichiarate come *static*, ci permettono di convertire il valore del tempo da *"tempo del processore"* in secondi. Della funzione *ToSeconds* esistono due versioni la prima prende come parametro un'istanza della classe *Stopwatch* ed effettua la conversione del valore restituito dal metodo *ClocksCount()* dell'oggetto passato la seconda invece effettua la conversione da un oggetto di tipo *clock_t*.
La classe non contiene altri metodi ed è pronta per essere utilizzata.

‹6› UTILIZZIAMO LA CLASSE

```
//Dichiaro e avvio il cronometro.
Stopwatch sw(true);

//compiamo qualche operazione
int dummy = 0;

for( int j = 0; j< 10000; j++)
 for( int i = 0; i< 100000; i++)
  dummy = j;

//Stampo il parziale ma non interrompo il conteggio

cout << Stopwatch::ToSeconds( sw )
             << endl;
for( int j = 0; j< 10000; j++)
 for( int i = 0; i< 100000; i++)
    dummy = i;
//Fermo il conteggio e stampo il tempo trascorso
cout << Stopwatch::ToSeconds( sw.Stop() )
<< endl;
```

Questa è una carrellata delle funzionalità della classe appena creata.

XSL; il trasformista per eccellenza

Una tecnica comoda, capace di trasformare qualunque documento XML in un altro formato. Facile da usare e utilizzabile da qualunque linguaggio. Scopriamola insieme!

Possiamo definire XSL come un insieme di istruzioni per trasformare un input in formato XML in vari tipi di output (altro XML, HTML, testo ecc...). Le istruzioni che occorrono per dare il via alla trasformazione sono anch'esse scritte in formato XML. XSL infatti non è altro che un'applicazione specializzata di XML, nata con lo scopo di gestire la trasformazione dei dati. XSL si basa sulla lettura dei dati da un fonte XML. Ma come fa a leggere i contenuti dal documento di origine? Attraverso un linguaggio di selezione dei nodi chiamato *XPath*.
Possiamo pensare a *XPath* come all'equivalente di SQL per un database, cioè ad una particolare sintassi per estrarre le informazioni da una fonte dei dati. *XPath* non fa parte in senso stretto di XSL anche perché viene integrato spesso in altre tecnologie come i parser basati su DOM, tuttavia nel corso dell'articolo tratteremo diffusamente di questo linguaggio perché è alla base del meccanismo di trasformazione.

HELLO WORLD XSL

Potevamo mancare all'appuntamento con il famoso *"Hello World"?* Ovviamente no! Vediamo quindi un semplice esempio di XSL in azione.Partiamo da un semplice file XML

```
<?xml version="1.0"?>
<helloworld>
   <titolo>Hello world</titolo>
   <messaggio>Benvenuti al corso XSL di
                              ioProgrammo!</messaggio>
</helloworld>
```

Se apriamo questo file in Internet Explorer o Firefox abbiamo una vista sulla struttura come quella in **Figura 1**.

Nella stessa directory poi creiamo il file XSL

```
<?xml version="1.0" ?>
<helloworld>
  <titolo>Hello world</titolo>
  <messaggio>Benvenuti al corso XSL di IOProgrammo!</messaggio>
</helloworld>
```

Fig. 1: ***Ecco come si presenta un file XML se letto direttamente dal browser***

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
        "http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/ ">
<html>
  <head>
  <title>Esempio Hello World</title>
  </head>
  <body>
    <h1><xsl:value-of select="helloworld/titolo"/></h1 >
    <div>
     <i>
      <xsl:value-of select="helloworld/messaggio"/>
     </i>
    </div>
   </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Adesso modifichiamo il file XML inserendo il riferimento al foglio di stile nel modo seguente:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href=
                              "helloworld.xsl"?>
<helloworld>
  <titolo>Hello world</titolo>
  <messaggio>Benvenuti al corso XSL di
                     IOProgrammo!</messaggio>
</helloworld>
```

Se riapriamo adesso il file XML nel browser,

visualizzeremo la trasformazione definita nel file XSL, **Figura 2**.

Hello world

Benvenuti al corso XSL di IOProgrammo!

Fig. 2: ***Il file XML trasformato con XSL***

Avrete notato le analogie tra questo procedimento e il modo di collegare un foglio di stile CSS ad una pagina HTML:

```
<link type="text/css" href="style.css">
```

Infatti il ruolo di XSL nei confronti del file XML è concettualmente molto simile a quello di un foglio di stile CSS verso una pagina HTML.

METODI DI COLLEGAMENTO TRA XML E FOGLIO DI STILE

I metodi di collegamento tra XML e foglio di stile sono:

1. **Inserimento di una processing-instruction nel file XML** - È il metodo che abbiamo visto nel nostro esempio. La una *processing-instruction* è *<?xml-stylesheet type ="text/xsl" href="..."?>*. Con questo metodo si affida la trasformazione all'engine contenuto nel browser.

2. **Richiamare la trasformazione da programma** - si tratta di scrivere le istruzioni per richiamare da programma la trasformazione del file XML in uno dei linguaggi che forniscono un engine.

Come esempio di questa seconda tecnica mostriamo come richiamare la trasformazione da uno script javascript in Internet Explorer:

```
<SCRIPT language = "javascript">
   function transform(){
      var srcTree = new ActiveXObject(
                        "Msxml2.DOMDocument.4.0");
      srcTree.async=false;
      srcTree.load("helloworld.xml");

      var xsltTree= new ActiveXObject(
                        "Msxml2.DOMDOCUMENT.4.0");
      xsltTree.async = false;
      xsltTree.load("helloworld.xsl");

      return srcTree.transformNode(xsltTree); }
</SCRIPT>
```

In una Web Application è anche possibile effettuare la trasformazione lato server con ASP, ASP.NET, PHP o JAVA per restituire al browser direttamente l'output. Qui vediamo un esempio in PHP:

```
<?
$xmlDocText=file_get_contents(realpath("helloworld.
                                            xml"));
$xslDocText=file_get_contents(realpath("helloworld.xsl"));
$xml = new DOMDocument;
$xml->loadXML($xmlDocText);
$xsl = new DOMDocument;
$xsl->loadXML($xslDocText);
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl);
echo($proc->transformToXML($xml));
?>
```

Il primo metodo sembra il più semplice ed intuitivo. In teoria per costruire un sito basterebbe basarsi sui documenti XML ai quali associare i file XSL. In pratica, come abbiamo detto, questo metodo viene utilizzato molto poco perché:

1. Non tutti i browser, specialmente quelli più vecchi, supportano la trasformazione. Si rischierebbe di rendere il sito inaccessibile ad una larga fetta di visitatori.

2. Non sempre, anzi pochissime volte, la sorgente XML è un file fisico, più spesso si tratta di uno stream magari proveniente da un database.

3. Non è detto che stiamo utilizzando XSL in ambiente Web; si può utilizzare XSL con profitto anche in programmi stand-alone.

Per le applicazioni web si fa quindi più spesso ricorso alla trasformazione lato server.

LA STRUTTURA DI BASE DEL FILE XSL

Un file XSL deve obbligatoriamente presentare gli elementi *stylesheet* e *template*.

```
<xsl:stylesheet
  id = ""
  extension-element-prefixes = ""
  exclude-result-prefixes = ""
  version = "1.0" xmlns:xsl=
            "http://www.w3.org/1999/XSL/Transform">
...
</xsl:stylesheet>
```

Vi sarà probabilmente capitato di sentir parlare di XSL e XSLT come sinonimi.
In realtà XSL (ovvero *Extensible Stylesheet Language*) sta ad indicare una famiglia di linguaggi che comprende:

XSLT - il linguaggio per la trasformazione dell'XML
XPATH - il linguaggio per la selezione di parti di un documento XML
FO – il linguaggio per la trasformazione in formati specifici (come PDF e SVG)

Questo corso sarà principalmente incentrato sui primi due linguaggi ed useremo il termine XSL (come avviene abitualmente) per indicare XSLT.
Ulteriori informazioni possono essere reperite sul sito www.w3.org/Style/XSL .

L'elemento *<xsl:stylesheet>* deve essere il primo nodo del documento. In esso sono contenuti tutti gli altri elementi XSL ed ha come attributi :

1. **id** - *opzionale* - L'identificativo unico del nodo;
2. **extension-element-prefixes** - *opzionale* - gli spazi dei nomi, separati da spazio bianco, da usare per richiamare le funzioni degli oggetti estesi (ci torneremo sopra più avanti);
3. **exclude-result-prefixes** - *opzionale* - gli spazi dei nomi, separati da spazio bianco, utilizzati che non devono essere riportati nell'output prodotto;
4. **version** - *obbligatorio* - la versione del linguaggio XSL, attualmente la "1.0";
5. **xmlns:xsl** - *obbligatorio* - lo spazio di nomi del prefisso *xsl:* , quello che contraddistingue gli elementi della sintassi XSL dagli altri utilizzati nell'output. Attualmente è sempre *http://www.w3.org/1999/XSL/Transform.*

STRUMENTI DI SVILUPPO

Se è vero che come editor per i file XSL il notepad può bastare è anche vero che in pratica l'editing con il semplice editor di testo si rivelerebbe una vera tortura. Esiste quindi una vasta gamma di editor che consentono l'editing corredato di *intellisense, sintax highlighting* e debug integrato. Il mio editor preferito è *XSELERATOR* prodotto da www.marrowsoft.com perché specializzato proprio per XSL.

L'elemento *stylesheet,* unico per ogni file, può contenere anche altri spazi di nomi oltre a quello obbligatorio che identifica *xsl,* questi spazi dei nomi possono essere quelli contenuti nel documento di origine o essere dei riferimenti ad oggetti di estensione, ad esempio:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:msxsl="urn:schemas-microsoft-com:xslt"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

Dove si dichiarano altri due namespace: *msxsl* e *fo.* Gli spazi dei nomi sono seguiti da un *uri* che è la dichiarazione di uno schema che deve essere gestito dall'engine che processa l'XSL. In questo caso, ad esempio, il namespace *msxsl* (che consente di aggiungere degli scripts nel file XSL) è correttamente gestito dal processore XSL di Microsoft mentre non lo è dagli altri.
Gli elementi template definiscono dei modelli di trasformazione da applicare a particolari nodi o contesti.

```
<xsl:template
  name= "nome" |  match = Pattern
  priority = number
  mode = "">
</xsl:template>
```

Per seguire un'analogia rispetto ai linguaggi di programmazione, possiamo pensare ai template come ai metodi, cioè a delle unità riutilizzabili.
In particolare si distinguono due tipi di template:

1. **I template legati agli elementi** - che cioè vengono eseguiti ogni volta che il processore incontra quel dato elemento nel file di origine.
2. **I template nominati** - più simili alle funzioni che vengono esplicitamente richiamate.

I template legati agli elementi hanno come attributi:

- **match** - *obbligatorio* - il percorso *Xpath* dell'elemento a cui sono collegati.
- **mode** - *opzionale* - un nome che permette di avere più template collegati allo stesso elemento e che compiono operazioni di trasformazione diverse, richiamabili ognuno con il proprio mode.
- **priority** - *opzionale* - un valore numerico che nel caso di due o più template associati allo stesso elemento determina quello da eseguire. In realtà questo attributo ha un impiego prettamente legato alla fase di debug perché consente di "nascondere" l'esecuzione di uno o più template, un po' come commentare un pezzo di codice per non farlo eseguire.

I template nominati hanno come attributo:

- **name** - *obbligatorio* - il nome con cui sono richiamabili.

I due tipi di template per essere applicati debbono essere richiamati. I template legati agli elementi vengono richiamati con l'istruzione

<*xsl:apply-templates*> mentre i template nominati con <*xsl:call-template*>. In realtà esiste un template che non deve essere esplicitamente invocato, il template:

```
<xsl:template match="/">
</xsl:template>
```

Cioè quello associato al pattern *"/"* ovvero all'intero documento di origine, un po' come il metodo *Main* di un programma. Questo template viene sempre eseguito.
Ma adesso vediamo subito di focalizzare questi concetti con degli esempi concreti.

APPLICAZIONE DI TEMPLATE LEGATI AGLI ELEMENTI

Ripartiamo dal foglio di stile *Helloworld* visto in precedenza e riscriviamolo utilizzando i template collegati agli elementi:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
            "http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
   <xsl:apply-templates select=
                "helloworld"></xsl:apply-templates>
</xsl:template>

<xsl:template match="helloworld">
<html>
   <head>
   <title>Esempio Hello World</title>
   </head>
   <body>
      <xsl:apply-templates select=
                  "titolo"></xsl:apply-templates>
      <xsl:apply-templates select=
                 "messaggio"></xsl:apply-templates>
   </body>
</html>
</xsl:template>

<xsl:template match="titolo">
   <h1><xsl:value-of select="."/></h1>
</xsl:template>

<xsl:template match="messaggio">
      <div>
         <i>
            <xsl:value-of select="."/>
         </i>
      </div>
</xsl:template>
</xsl:stylesheet>
```

Viene dapprima eseguito il template *match="/"* che, con l'istruzione *apply-templates* associa il nodo *helloworld* del documento XML al template che lo gestisce *(template match="helloworld")*, quest'ultimo, a sua volta, oltre a scrivere dell'output richiama l'associazione ai rispettivi template per i sottonodi titolo e messaggio che definiscono la formattazione del testo contenuto. Un po' come dire:

1. Leggi il documento di origine e vai al nodo *helloworld*.
2. Sei adesso sul nodo *helloworld* qui scrivi un po' di codice di output e vai al nodo *titolo*.
3. Sei adesso sul nodo *titolo* esegui l'output definito nel template associato.
4. Vai al nodo *messaggio*.
5. Sei adesso sul nodo *messaggio* esegui l'output definito nel template associato.
6. Sei tornato al nodo *helloworld* scrivi il rimanente output ed esci.

L'istruzione *apply-templates,* ha come attributo *select. Select* è uno degli attributi più importanti di XSL e consente di puntare ad un determinato elemento del file di origine attraverso la sintassi di *XPath.* Tratteremo più diffusamente di *select* e di *XPath* nella seconda puntata del corso.

APPLICAZIONE DI TEMPLATE NOMINATI

Vediamo adesso lo stesso foglio di stile scritto utilizzando gli elementi nominati invece di quelli legati agli elementi:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
            "http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html>
   <head>
   <title>Esempio Hello World</title>
   </head>
   <body>
   <xsl:call-template name="write-titolo">
                              </xsl:call-template>
   <xsl:call-template name="write-messaggio">
                              </xsl:call-template>
   </body>
</html>
</xsl:template>
```

MOTORI XSL

Esistono numerose librerie integrate o integrabili nei vari linguaggi di programmazione. Un sito molto ricco di informazioni in questo senso è www.topxml.com, una vera e propria miniera di informazioni e tutorial su XSL e tecnologie correlate.

```
<xsl:template name="write-titolo">
   <h1><xsl:value-of select="helloworld/titolo"/></h1>
</xsl:template>

<xsl:template name="write-messaggio">
   <div>
      <i>
        <xsl:value-of select="helloworld/messaggio"/>
      </i>
   </div>
</xsl:template>

</xsl:stylesheet>
```

Possiamo vedere come sia diverso il modo di richiamare il template: *call-template*. Infatti non si basa più su una selezione di un elemento del file XML di origine, bensì sul nome stesso del template.
I template di questo tipo sono più simili a metodi e funzioni dei linguaggi di programmazione tradizionali, tanto che possono accettare anche parametri e non essere legati alla lettura del file XML originale.
Anche sui template nominati torneremo nella seconda puntata del corso quando prenderemo in esame:

- Parametri, variabili e loro *scope*
- Elementi per l'iterazione *<xsl:for-each>*
- Strutture condizionali *<xsl:if>* e *<xsl:when>*

UN CATALOGO DI LIBRI

L'obiettivo del nostro esempio è quello di utilizzare un file XML contenente la rappresentazione di un catalogo di libri per creare una pagina web che lo mostri in una semplice tabella.
Partiamo quindi dal file xml di esempio tratto dalla documentazione Microsoft

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating
                                    applications
      with XML.</description>
   </book>
...
</catalog>
```

Quindi creiamo una pagina ASP.NET (nel CD il file è chiamato *esempio1.aspx*) di cui qui sotto riportiamo solamente il codice VB.NET utilizzato per leggere i dati dal file xml e costruire il codice HTML:

```
 <script language="VB" runat="server">
    Public Sub Page_Load()
 Dim  docParser as new XmlDocument
 Dim  xmlFilepath as String = mapPath("esempio1.xml")
 Dim  s as String = ""
 docParser.Load(xmlFilepath)
 Dim root as XmlElement = docParser.DocumentElement
      s= "<table>"
      s &= "<tr>"
      s &= "<th class=""table-head"">ID</th>"
      s &= "<th class=""table-head"">Autore</th>"
      s &= "<th class=""table-head"">Titolo</th>"
      s &= "<th class=""table-head"">Genere</th>"
      s &= "<th class=""table-head"">Prezzo</th>"
      s &= "<th class=""table-head"">Data</th>"
      s &= "<th class=""table-head"">Descrizione</th>"
      s &= "</tr>"
      for each book as XmlElement in
                              root.SelectNodes("book")
      Dim bookId as string = book.getAttribute("id")
      Dim author as string =
                        book.Item("author").innerText
      Dim title as string = book.Item("title").innerText
      Dim genre as string =
                         book.Item("genre").innerText
      Dim price as string = book.Item("price").innerText
      Dim publish_date as string = _
      book.Item("publish_date").innerText
      Dim description as string =
                    book.Item("description").innerText
      s &= "<tr>"
      s &= "<td  class=""table-cell""><b>" & bookId
                                      & "</b></td>"
      s &= "<td  class=""table-cell"">" & author &
                                           "</td>"
      s &= "<td  class=""table-cell"">" & title &
                                           "</td>"
      s &= "<td  class=""table-cell"">" & genre &
                                           "</td>"
      s &= "<td class=""table-cell""
            bgcolor=""#DEDEDE"" align=""right"">"_
      & price & "</td>"
      s &= "<td  class=""table-cell"">" & cType(
       publish_date,datetime).ToString("dd/MM/yyyy")
                                         & "</td>"
      s &= "<td  class=""table-cell"">" & description
                                         & "</td>"
      s &= "</tr>"
      next
```

```
        s &= "</table>"
text1.Text=s 'text1 è il controllo Literal che ospita
                                              l'output
      End Sub
   </script>
```

Il risultato sarà una tabella HTML con l'elenco dei dati contenuti nel file XML.
Qui abbiamo utilizzato il parser di .NET con VB.NET ma la logica non cambia con altri linguaggi come ad esempio PHP o PERL. Il problema è, in tutti i casi, la commistione tra la logica di formattazione dell'output, la sintassi del linguaggio e la lettura dei dati XML.
È certo che un codice di questo tipo, "spalmato" in progetti di dimensione medio-grande, produrrà presto una notevole difficoltà di gestione: cosa succede se voglio cambiare la formattazione di una cella?
O se cambio anche di poco la struttura dei dati?
Trovare il punto dove applicare i cambiamenti in decine di files si rivelerà un'impresa non di poco conto!
La situazione poi si aggrava se, com'è normale, le figure professionali che si occupano dell'HTML sono diverse dai programmatori.
Qui l'XSL rappresenta il nostro "asso nella manica".
Ecco come si presenta il file XSL che contiene la tutta la logica di trasformazione dai dati XML alla pagina XML (nel CD *esempio2.xsl*):

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
              "http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
  <xsl:template match="/">
  <table>
     <tr>
        <th class="table-head">ID</th>
        <th class="table-head">Autore</th>
        <th class="table-head">Titolo</th>
        <th class="table-head">Genere</th>
        <th class="table-head">Prezzo</th>
        <th class="table-head">Data</th>
        <th class="table-head">Descrizione</th>
     </tr>
        <xsl:for-each select="/*/book">
     <tr>
        <td class="table-cell">
        <b><xsl:value-of select="@id"/></b></td>
        <td class="table-cell">
        <xsl:value-of select="author"/></td>
        <td class="table-cell">
        <xsl:value-of select="title"/></td>
        <td class="table-cell">
           <xsl:value-of select="genre"/></td>
           <td class="table-cell" align="right" bgcolor=
                                           "#DEDEDE">
           <xsl:value-of select="price"/></td>
           <td class="table-cell">
              <xsl:value-of select="publish_date"/></td>
              <td class="table-cell">
              <xsl:value-of select="description"/></td>
        </tr>
     </xsl:for-each>
     </table>
  </xsl:template>
</xsl:stylesheet>
```

Come vedete, è molto chiaro. Codice HTML standard con alcuni tags particolari, niente logica di programmazione. Comprensibilissimo anche da parte di un web-designer.
Non ci soffermeremo adesso sulla sintassi XSL, ma vediamo invece come si presenta adesso il codice VB.NET, necessario per "catturare" l'output della presentazione definito nell'XSL (nel CD è il file *esempio2.aspx*):

```
  <script language="VB" runat="server">
  Public Sub Page_Load()
        Dim xslProcessor As New Xsl.XslTransform
        Dim xmlFilepath As String = mapPath(
                                     "esempio1.xml")
        Dim xmlDoc As New XPath.XPathDocument(
                                        xmlFilepath)
        Dim xslFilepath As String = mapPath(
                                     "esempio2.xsl")
        Dim result As String
        xslProcessor.Load(xslFilepath)
        Dim out As New IO.MemoryStream
        xslProcessor.Transform(xmlDoc, Nothing, out)
        out.Position = 0
        Dim rd As New IO.StreamReader(out)
        result = rd.ReadToEnd
        rd.Close()
   text1.Text=result 'text1 è il controllo Literal che
                                     ospita l'output
   End Sub
  </script>
```

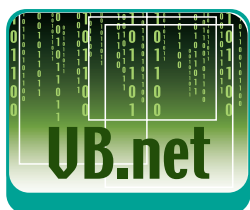
Per concludere, si instanzia l'oggetto che eseguirà la trasformazione *(XslTransform)*, gli passiamo i riferimenti al file XSL ed eseguiamo la trasformazione, con il metodo *Transform* passando come parametro il riferimento al file XML ed uno *Stream* di cui leggeremo poi il contenuto.
Il metodo di trasformazione varia da linguaggio a linguaggio, tuttavia il concetto di base resta il solito: la completa separazione tra logica di programmazione e presentazione! XSL e XSLT

Francesco Smelzo

GLI ESEMPI DEL CORSO

Gli esempi di codice che troverete in questo corso sono stati scritti in diversi linguaggi di programmazione. Questo a sottolineare che l'interpretazione delle istruzioni XSL può avvenire nei più diversi contesti applicativi. Ciò non toglie che ogni specifica implementazione aggiunga estensioni proprietarie alla sintassi XSL standard (anche se, ovviamente, l'utilizzo di estensioni proprietarie limita fortemente la portabilità del codice).

La Gestione degli Errori

In questo appuntamento impareremo come costruire applicazioni a prova di errore! Gestiremo tutte le possibili combinazioni nel tentativo di produrre software sempre più affidabile

Qualsiasi programmatore, anche il più esperto, prima o poi commette un errore nella stesura di un programma. Sapere quali tipi d'errore si possono verificare e come correggerli, consente di ridurre notevolmente il tempo necessario allo sviluppo di una applicazione.
Gli errori possono essere divisi in tre categorie principali

- **Errori di sintassi** (e di compilazione)
- **Errori di run-time** (e di esecuzione)
- **Errori logici** (e di progettazione)

GLI ERRORI DI SINTASSI

Gli errori di sintassi si verificano quando il linguaggio non è in grado di capire il codice appena scritto, perché un comando è digitato in maniera errata, le istruzioni sono incomplete, oppure sono scritte in un ordine non previsto. Ad esempio scrivendo

```
din MiaVariabile as Integer
```

Invece di

```
dim MiaVariabile as Integer
```

VB .Net non sarà in grado di interpretare l'istruzione causando l'arresto del programma.
VB dispone di un meccanismo molto sofisticato per il controllo della sintassi, verifica in tempo reale ciò che viene scritto e segnala subito eventuali errori, permettendone l'individuazione.
Nel caso del codice errato dell'esempio precedente, l'ambiente di sviluppo risponde con le seguenti azioni:

- Sottolinea l'istruzione errata;
- Nel riquadro delle attività segnala i possibili errori;
- Mostra una descrizione dell'errore al passaggio del mouse sull'istruzione.

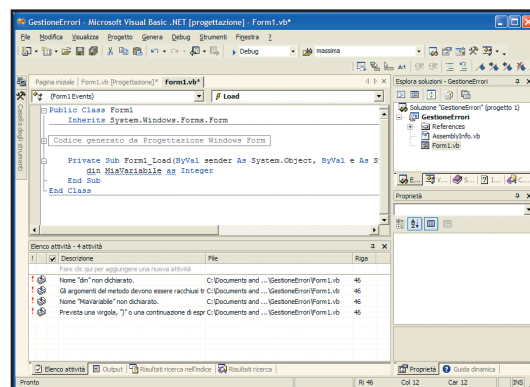



***Fig. 1:** Il codice non corretto viene evidenziato in fase di editing*

Uno degli errori di sintassi più comune, è la scrittura errata del nome di una variabile. Poiché non è consentito utilizzare variabili non dichiarate, non appena scriviamo un nome errato, tale nome sarà immediatamente evidenziato dall'ambiente di sviluppo. Unicamente dopo aver corretto tutti gli errori di sintassi, VB permetterà al programma di essere eseguito, a questo punto ci dovremmo preoccupare solo (per modo di dire) degli errori logici e di run-time.

ERRORI DI RUN-TIME

Gli errori di run-time si riscontrano esclusivamente durante l'esecuzione del programma, e possono verificarsi anche quando la sintassi del codice è corretta. Un classico errore di run-time si può verificare se utilizziamo un operazione di divisione:

```
VelocitaMedia = KmPercorsi \ OreTrascorse
```

L'istruzione è scritta correttamente e restituisce sempre il risultato voluto, salvo che l'utente distratto inserisca il valore zero per *OreTrascorse,* nel qual caso il compilatore genera un errore di *Run-Time* e solleva un'eccezione (*DivideByZeroException,* Tentativo di divisione per zero). Per evitare gli errori di run-time, dobbiamo utilizzare la gestione degli errori in modo da rilevare e gestire ogni possibile errore, come vedremo in seguito.

ERRORI LOGICI E DI PROGETTAZIONE

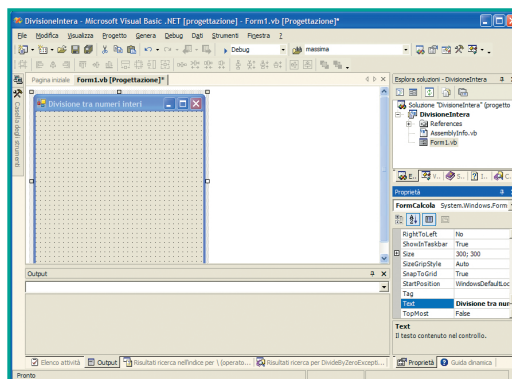
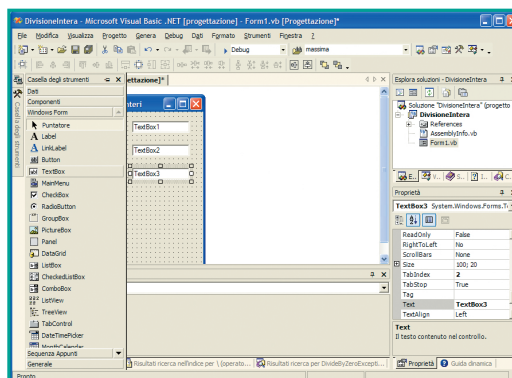
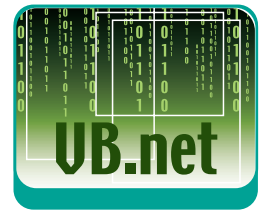
Gli *Errori logici* sono gli errori più difficili da individuare. Sono generati quando l'esecuzione dell'applicazione è diversa da quella prevista. Si può scrivere una applicazione sintatticamente corretta e gestire tutti i possibili errori di run-time, ma si possono dare, ad esempio, dei comandi che eseguiti prima di altri possono causare un comportamento inaspettato del programma. Per loro natura sono molto difficili da individuare, l'unico modo per ridurli è di prevenirli con una robusta analisi del progetto. Per scovare un errore logico, VB.Net mette a disposizione dei potenti strumenti di debug.

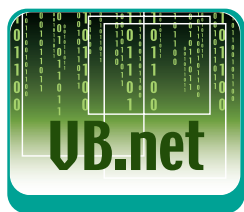
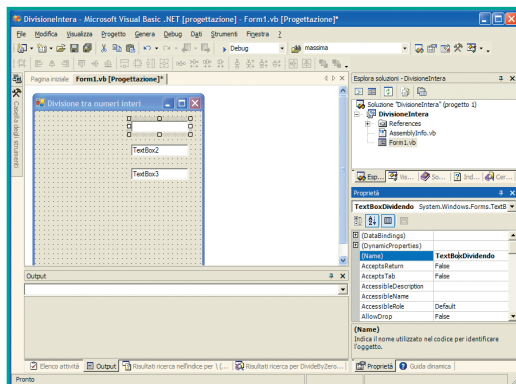
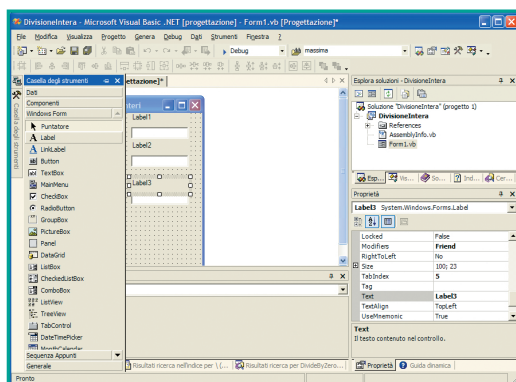
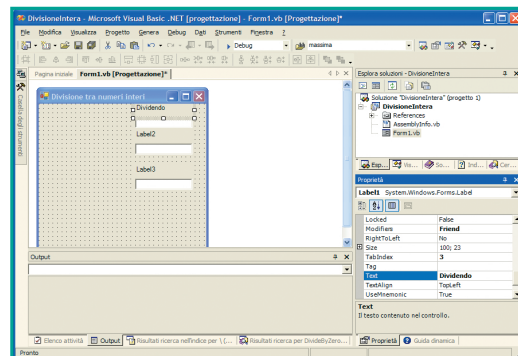
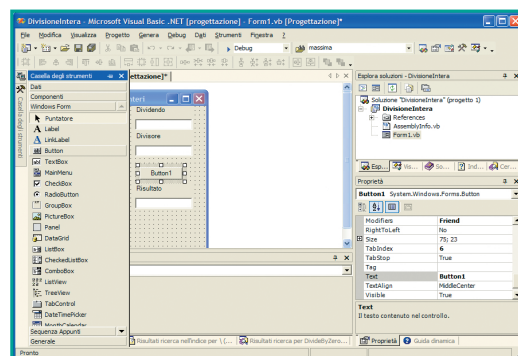
PROGETTARE UN GESTORE DI ERRORI

In un *gestore di errori* si devono prevedere le istruzioni per intercettare gli errori, ed il codice necessario per la risposta agli errori generati. Di solito è necessario presumere la possibilità di errore in qualsiasi istruzione VB, sempre che non si sia assolutamente certi del contrario. In VB.Net è possibile gestire gli errori, e l'oggetto *Eccezione* creato dall'errore, in modalità strutturata o non strutturata. La gestione delle eccezioni strutturata, consiste nell'utilizzo di un meccanismo di gestione basato su una struttura di controllo (La struttura *Try...Catch*) che verifica blocchi di codice specifici e, se viene generata un'eccezione, adatta il codice di gestione alla tipologia di eccezione verificatasi. Tale metodo consente al codice di distinguere i diversi tipi di errore e di reagire secondo i casi. Nella gestione delle eccezioni non strutturata, viene utilizzata un'istruzione *On Error* all'inizio del codice per gestire tutte le eccezioni. È possibile utilizzare contemporaneamente la gestione delle eccezioni strutturata e non strutturata, a patto che non siano nella stessa procedura. Praticamente se utilizziamo un'istruzione *On Error,* non è possibile inserire un'istruzione *Try...Catch* nella stessa procedura. I moderni dettami della programmazione sconsigliano la tipologia di gestione non strutturata, che era però l'unica tipologia presente in VB6, e pertanto verrà descritta brevemente poiché ci potremmo trovare ad esaminare del codice importato dalle precedenti versioni di Visual Basic.

IL PROGETTO DI ESEMPIO

Per gli esempi descritti nell'articolo, creiamo una nuova soluzione, con la solita procedura vista nei numeri precedenti, dal nome *"DivisioneIntera"* e disegniamo una semplice interfaccia utente

1 Selezioniamo la finestra *Form1* e, se non è già visualizzata, apriamo la finestra delle proprietà. Dalla finestra delle proprietà selezioniamo la proprietà *Name* e cambiamo subito il nome in: *FormCalcola.* Selezioniamo la proprietà Text e modifichiamo il testo visualizzato nella barra del titolo in: *Divisione tra numeri interi.*

2 Selezioniamo per tre volte un controllo *TextBox* dalla casella degli strumenti (nella sezione *Windows Form*) e disegniamo i tre controlli sulla form.

3 Selezioniamo il primo *Textbox* e, dalla finestra delle proprietà, evidenziamo la proprietà

GLI OPERATORI

Negli esempi di codice abbiamo utilizzato l'operatore \ (operatore di divisione intera) che permette di dividere due numeri restituendo un valore Integer, invece del classico operatore di divisione / che restituisce un risultato in virgola mobile. La scelta è stata fatta perché in VB.Net 2003, dividendo un valore in virgola mobile per zero, si ottiene un risultato: infinito positivo, infinito negativo o un valore non numerico in base alle regole aritmetiche, e pertanto le operazioni a virgola mobile non generano mai un'eccezione.

Name per cambiare il nome in: *TextBoxDividendo.* Evidenziamo la proprietà *Text* e cambiamo il testo nella stringa vuota.
Allo stesso modo: selezioniamo il secondo *TextBox* e variamo la proprietà *Name* in: *TextBoxDivisore* e la proprietà *Text* a vuoto; selezioniamo il terzo *TextBox* e variamo la proprietà *Name* in: *TextBoxRisultato* e la proprietà *Text* a vuoto.

4 Selezioniamo per tre volte un controllo *Label* dalla casella degli strumenti e disegniamo i tre controlli sulla form in corrispondenza dei tre *TextBox* disegnati in precedenza.

5 Selezioniamo la prima Label e, dalla finestra delle proprietà, evidenziamo la proprietà *Text* per cambiare il testo visualizzato in: *Dividendo.* Allo stesso modo: selezioniamo la seconda Label e variamo la proprietà *Text* in: *Divisore;* selezioniamo la terza Label e variamo la proprietà *Text* in: *Risultato*

6 Selezioniamo un controllo *Button* dalla casella degli strumenti e disegniamolo sulla form.

7 Selezioniamo il controllo *Button* e, dalla finestra delle proprietà, modifichiamo la proprietà *Name* in: *ButtonCalcola* e la proprietà *Text* in: *Calcola*
Nei due TextBox d'input (*TextBoxDividendo* e *TextBoxDivisore*), l'utente dovrà inserire i valori del dividendo e del divisore. Quando l'utente clicca con il mouse sul bottone, sarà lanciata la procedura di evento *ButtonCalcola_Click* che mostrerà

L'OGGETTO EXCEPTION

L'oggetto *Exception* contiene informazioni su qualsiasi errore di runtime rilevato. Ogni volta che viene generata un'eccezione, vengono impostate le proprietà dell'oggetto *Err* e viene creata una nuova istanza dell'oggetto *Exception.* L'oggetto *Exception* espone le seguenti proprietà

HelpLink può contenere un collegamento al file della *Guida* che indirizza l'utente a ulteriori informazioni relative all'eccezione.

Hresult rappresenta un valore numerico a 32 bit assegnato all'eccezione. Contiene tre campi: un codice di gravità, un codice di servizio e un codice di errore.

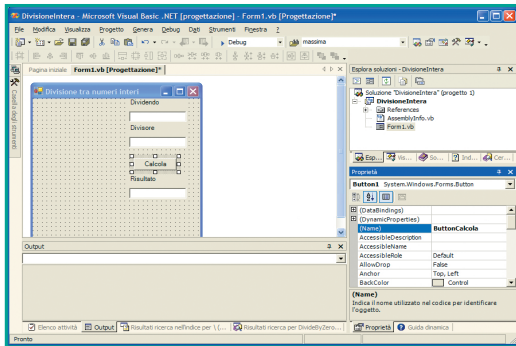
InnerException restituisce un oggetto *Exception* che rappresenta l'istanza dell'eccezione corrente. È possibile nidificare le eccezioni, in questo caso la proprietà *InnerException* fornisce l'accesso all'eccezione più interna.

Message contiene una stringa corrispondente al messaggio di testo che rappresenta la descrizione dell'eccezione (simile ad *Err.Description*).

Source imposta o restituisce una stringa contenente il nome dell'oggetto che ha generato l'eccezione (simile ad *Err.Source*).

StackTrace contiene una traccia dello stack, che consente di determinare il punto del codice in cui si è verificato l'errore.

TargetSite restituisce il nome del metodo che ha generato l'eccezione corrente. Tutte le proprietà, a eccezione di *Source* e *HelpLink* sono di sola lettura.

nel TextBox di Output *(TextBoxRisultato)* il risultato dell'operazione.

LA GESTIONE DEGLI ERRORI NON STRUTTURATA

Per segnalare il punto in cui VB deve iniziare ad intercettare gli errori, dobbiamo scrivere l'istruzione:

```
On Error GoTo riga
```

dove riga rappresenta l'etichetta che indica la riga iniziale del codice necessario alla gestione degli errori. Questa istruzione rimane attiva finché è attiva la procedura che la contiene, e vale a dire fino a quando non viene eseguita un'istruzione: *Exit Sub, Exit Function, Exit Property, End Sub, End Function* o *End Property.*
In una procedura è possibile attivare un solo punto di intercettazione alla volta, è però possibile creare più punti alternativi ed attivare quello desiderato secondo i casi. È inoltre possibile disattivare l'intercettazione degli errori utilizzando una particolarità dell'istruzione *On Error*, e vale a dire *On Error GoTo 0* (zero).
Se in una procedura si verifica un errore, in un'istruzione successiva ad *On Error,* l'applicazione salta alla riga di codice successiva alla riga in cui è stata definita l'*etichetta.*
Per scrivere il codice è sufficiente fare doppio click sul controllo *Button.* Con questa operazione si aprirà la finestra del codice con il cursore posto all'interno della procedura di evento *ButtonCalcola_Click,* il codice da inserire è il seguente

```
On Error GoTo GestoreErrori
        TextBoxRisultato.Text =
                        CInt(TextBoxDividendo.Text) \
CInt(TextBoxDivisore.Text)
Exit Sub
GestoreErrori: Select Case Err.Number
   Case 11
   MessageBox.Show("Errore: Tentativo di divisione
```

```
                                        per zero")
    Case Else
       MessageBox.Show("Errore non previsto num. " &
                                        Err.Number)
   End Select
```

L'istruzione *On Error GoTo GestoreErrori* indica il punto in cui inizia la gestione degli errori e che da quel momento in poi, in caso di errore si deve smettere di eseguire il codice e saltare alle istruzioni immediatamente successive all'etichetta *GestoreErrori.*
Il codice che effettua fisicamente il calcolo è il seguente

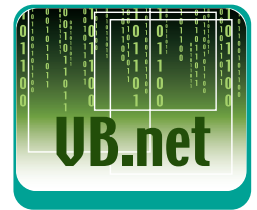
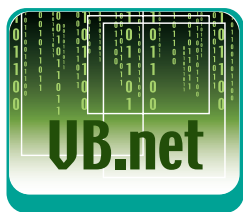
```
TextBoxRisultato.Text = CInt(TextBoxDividendo.Text)
                              \ CInt(TextBoxDivisore.Text)
Exit Sub
```

LA TECNOLOGIA INTELLISENSE

Uno dei punti di forza dell'ambiente di sviluppo di VB è sempre stata la tecnologia IntelliSense, che aiuta a evitare gli errori di sintassi. IntelliSense dispone di numerose funzioni quali ad esempio un elenco a discesa di membri delle classi e delle strutture dei namespace. Ciò offre due importanti vantaggi: non dobbiamo ricordare tutti i membri disponibili per la classe, poiché è sufficiente scorrere l'elenco e trovare il membro che dobbiamo utilizzare, inoltre si prevengono gli errori di sintassi, poiché non dobbiamo digitare il nome del membro e non rischiamo di commettere errori di digitazione. Per selezionare il membro desiderato, si deve premere il tasto Tab o Invio, oppure fare doppio clic sul membro stesso.

Prima dell'etichetta di riga si deve prevedere un'istruzione *Exit Sub*, in caso contrario, il codice successivo viene eseguito anche se non viene generato alcun errore.
L'etichetta di riga viene specificata da un nome descrittivo *(GestoreErrori)* seguito dai due punti, per contrassegnare l'inizio della routine di gestione degli errori (l'etichetta può essere anche un numero, ma per la leggibilità del codice è sempre meglio usare un nome descrittivo). Infine è necessario scrivere il codice per gestire gli errori

```
Procedura di gestione degli errori
    Select Case Err.Number
            Case 11
                MessageBox.Show("Errore: Tentativo di
                                   divisione per zero")
            Case Else
                MessageBox.Show("Errore non previsto
                                   num. " & Err.Number)
     End Select
```

La procedura di gestione degli errori contiene,

un'istruzione *Case* necessaria per stabilire quali sono i possibili errori e fornire per ciascuno di essi una risposta adeguata, nel nostro caso ci limitiamo ad avvisare l'utente di un errata divisione per zero. È comunque consigliabile inserire sempre l'istruzione *Else* o *Case Else* per fornire un'opzione per la gestione di errori imprevisti, proviamo ad esempio ad inserire un valore non numerico nei campi ed osservare cosa succede.
L'oggetto *Err* contiene informazioni concernente gli errori di Run-time. La proprietà *Number* dell'oggetto *Err* specifica un codice numerico che rappresenta l'ultimo errore di run-time generato, mentre la proprietà *Description* fornisce la descrizione di tale errore.

LA GESTIONE DEGLI ERRORI STRUTTURATA

Il procedimento per la generazione e l'intercettazione degli errori, utilizzato in VB.NET 2003, si basa sulle eccezioni.
Se durante la normale esecuzione del codice si verifica un errore, VB interrompe il flusso di codice e crea un oggetto *Exception*. Il programma tenta di trovare un gestore di eccezioni (un blocco di codice che indichi come agire in conseguenza dell'errore) per riprendere il flusso con l'esecuzione di questo blocco, e se non lo trova interrompe l'esecuzione dell'applicazione. Si dice che il codice solleva un'eccezione che qualche altro pezzo di codice dovrà intercettare.
Un'eccezione può essere intercettata nella stessa procedura nella quale si verifica l'errore, se ciò non accade, l'eccezione viene sollevata alla procedura chiamante. Se la procedura chiamante non contiene un gestore di errori, allora l'eccezione risale lo stack delle chiamate fino a quando non individua una procedura in grado di intercettarla.
Il gestore di errori può esaminare le proprietà, oppure invocare i metodi, dell'oggetto *Exception*, in modo da poter eseguire tutte le operazioni necessarie alla risoluzione del problema verificatosi come: fornire all'utente un messaggio di errore, cancellare e risolvere l'errore in maniera tacita oppure consentire all'utente di correggere l'errore e di procedere normalmente. Esaminiamo ora in dettaglio, i costrutti necessari ad una corretta gestione degli errori strutturata.

NOTE SULLE VERSIONI

La descrizione dell'errore varia secondo la versione localizzata di VB, per questo motivo il controllo sul tipo di errore deve essere fatto necessariamente su *Err.Number* e non su *Err.Description.*

IL GESTORE DI ERRORI TRY...CATCH...FINALLY

Il gestore di errori *Try...Catch...Finally* verifica una porzione di codice ed indica all'applicazione come gestire le varie tipologie di errore verificabili.
Ognuno dei tre componenti svolge un ruolo specifico:

- La parte di codice tra l'istruzione *Try* e la prima istruzione *Catch*, contiene la porzione di codice che potrebbe generare un eccezione. Nel caso in cui il codice solleva un'eccezione il flusso di codice passa alla prima clausola *Catch* che identifica l'eccezione specifica.

- Nel blocco *Catch* è possibile esaminare le proprietà dell'oggetto eccezione e decidere come reagire all'errore. Il blocco *Catch...As* verifica un oggetto eccezione del tipo indicato, e specifica il corrispondente codice da eseguire. Una clausola *Catch* senza un blocco *As* determina una risposta a qualsiasi eccezione.
  Pertanto il codice potrebbe contenere una serie di istruzioni *Catch... As* specifiche, ognuna delle quali controlla e fornisce una risposta ad una specifica eccezione, seguita da un blocco *Catch* generico che reagisce a qualsiasi eccezione non rilevata dalle istruzioni precedenti.

- Il blocco *Finally* contiene il codice da eseguire, indipendentemente dal fatto che si sia sollevata un'eccezione nel blocco *Try*. Il blocco di codice tra *Finally* ed *End Try* viene eseguito anche se si esce dal blocco *Try...End Try* a causa di un'istruzione *Exit Try* o *Exit Sub*. Possiamo utilizzare questa opportunità, per eseguire, ad esempio, le parti di codice addette alle attività di pulitura quali la chiusura dei file o la chiusura di un *RecordSet*.

È possibile uscire da una struttura *Try.. .End Try* in qualsiasi istante invocando *End Try*.
Utilizzando il gestore di errori *Try...Catch* possiamo riscrivere l'esempio precedente:

```
Private Sub ButtonCalcola_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                        Handles ButtonCalcola.Click
    Try
        TextBoxRisultato.Text =
                    CInt(TextBoxDividendo.Text) \
                       CInt(TextBoxDivisore.Text)
    Catch ex As DivideByZeroException
        MessageBox.Show("Errore: Tentativo di
                                divisione per zero")
    Catch ex As Exception
        MessageBox.Show("Errore non previsto:
                                   " & ex.Message)
```

```
    End Try
End Sub
```

LA CLAUSOLA CATCH

Per la clausola *Catch,* possiamo utilizzare tre tipi di sintassi: Catch, Catch...As e Catch...As...When.

- La clausola Catch senza la parola chiave As o When consente al blocco di istruzioni associate di gestire qualsiasi tipo di eccezione.
- Le clausole Catch...As consentono di rilevare un'eccezione specifica e permettono al blocco di istruzioni associate, di indicare all'applicazione il tipo di risposta necessaria.
- La clausola Catch...As...When consente di testare un'ulteriore condizione che deve necessariamente valere True perché il blocco Catch corrispondente venga eseguito.
  In generale, è possibile ottenere lo stesso comportamento utilizzando un costrutto If.. .Elself all'interno di un blocco Catch, ma la clausola When permette una miglior organizzazione del codice per la gestione degli errori.

Dobbiamo porre particolare attenzione all'ordine in cui si scrivono le clausole Catch...As, poiché VB.Net confronta il tipo di oggetto eccezione con le espressioni contenute nelle clausole, nell'ordine con cui queste appaiono, esegue il codice della clausola corrispondente ed esce dal gestore degli errori senza esaminare le altre clausole. Pertanto, nelle clausole Catch, si devono verificare prima le eccezioni specifiche e poi quelle generiche.
Ad esempio, il blocco Catch per DivideByZeroException non dovrebbe mai seguire il blocco Catch per un oggetto ArithmeticException più generico che lo comprende.

LA PAROLA CHIAVE FINALLY

In alcuni casi, potrebbe essere necessario prevedere del codice che deve essere eseguito indipendentemente dal fatto che si verifichi o meno un errore.
È il tipico caso del codice che deve liberare risorse, come chiudere un file o fare il Dispose di un oggetto. In casi come questi, per eseguire del codice senza condizioni possiamo servirci della clausola Finally.
La porzione di codice contenuta tra le parole chiave Finally ed End Try viene eseguita:

- indipendentemente dal fatto che il blocco Try abbia sollevato o meno un'eccezione
- nei casi in cui il codice di un blocco Catch solleva un'eccezione
- se si esce dal blocco Try...End Try a causa di un'istruzione Exit Try.

Dobbiamo fare attenzione, a che non si verifichino errori durante l'elaborazione del codice Finally perché in questo caso VB esce immediatamente dal blocco e l'eccezione viene notificata al chiamante. Se esiste la possibilità che le istruzioni all'interno del blocco Finally possano generare un errore, si può utilizzare una struttura Try...End Try annidata all'interno del blocco.

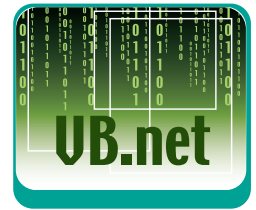
LA PAROLA CHIAVE THROW

In alcuni casi può risultare utile generare un errore per indicare alle procedure chiamanti che si è verificata un'eccezione, per questo si deve utilizzare la parola chiave Throw che genera un'eccezione dal blocco corrente.
In VB 6 è possibile generare un errore utilizzando il metodo Err.Raise, e poiché VB .NET 2003 gestisce ancora l'oggetto Err, ogni codice basato sul metodo Raise continuerà a funzionare come prima. Ciò nonostante, per conformità con il nuovo meccanismo delle eccezioni, è sempre consigliabile sollevare le eccezioni utilizzando il comando Throw.
L'istruzione Throw accetta come argomento soltanto l'oggetto eccezione che viene sollevato. Per generare un'eccezione si deve creare un oggetto di tipo eccezione ed impostarne le relative proprietà, nella maggior parte dei casi è possibile creare un oggetto eccezione e sollevarlo in un'unica istruzione:

```
Throw New DivideByZeroException()
```

CONCLUSIONI

Anche i programmatori più bravi commettono prima o poi un errore, basta guardare le varie Service Pack che rilascia periodicamente la Microsoft (e non solo lei) per rimediare agli errori dei propri programmi di punta, l'importante è non lasciarsi scoraggiare e conoscere gli strumenti per snidarli.

Luigi Buono

NOTA

CONSIGLI UTILI

È buona norma avere una clausola *Catch* che verifichi l'oggetto *System.Exception.* L'oggetto *System.Exception* contiene qualsiasi tipo di eccezione, perciò questa clausola deve essere scritta nell'ultimo blocco in assoluto perché i blocchi successivi certamente non verranno mai eseguiti.

Macchine Virtuali Soluzione per i test!

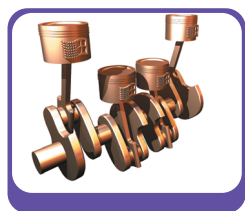
Utilizziamo VirtualPC per installare su una sola macchina fisica molti sistemi operativi. Lo scopo è simulare il comportamento delle nostre applicazion in ambienti eterogenei

Virtual PC è un software che consente di creare macchine virtuali dotate di un sistema operativo proprio basandosi su un'unica macchina fisica. Le macchine virtuali gireranno in una finestra come una normale applicazione di Windows, ma avranno un proprio OS e si comporteranno di conseguenza.
Sarà così possibile avere sul proprio PC una macchina fisica con Windows XP installato, e più macchine virtuali con altri OS installati ad esempio Windows 2000, 98 oppure Linux. Lo scopo di questo è evidente: riuscire ad effettuare dei test di comportamento dei propri programmi su sistemi operativi eterogenei. Una versione demo valida per 45 giorni di Virtual PC è disponibile all'indirizzo: *http://www.microsoft.com/virtualpc*. Virtual PC funziona su Windows 2000 o superiori e può ospitare moltissimi tipi di sistemi operativi diversi, purché funzionanti su piattaforma x86 Intel o AMD.

Fig. 1: **Quattro sessioni di macchine virtuali salvate in differenti stati**

I sistemi operativi vengono ospitati in PC "virtuali" (*Virtual Machine* o VM d'ora in poi) all'interno del PC vero e proprio, ognuno con un proprio BIOS, una scheda grafica, uno o più hard disk, porte di rete, seriali, parallele, floppy, cd. Solo due cose sono "fisicamente" condivise con il PC reale: processore e memoria. Nel senso che i PC virtuali sfruttano il processore e la memoria del PC ospite, e quindi per poter lavorare in maniera decente, bisogna avere molta RAM e processori potenti. Con Virtual PC è possibile eseguire più VM contemporaneamente, e anche in questo caso le richieste di memoria e processore salgono esponenzialmente. Con un Pentium IV da almeno 2Ghz e 768Mb di RAM si riesce a lavorare tranquillamente, ma naturalmente più ce n'è meglio è!

CONFIGURIAMO VIRTUAL PC PER LA TASTIERA ITALIANA

Prima di cominciare consiglio vivamente di cambiare il tasto Host. Il tasto Host viene usato assieme a Canc per inviare la sequenza di Ctrl-Alt-Canc nelle VM, e assieme ad Enter per entrare/uscire dalla modalità full-screen. Di default viene mappato su Alt-Gr, ma conviene spostarlo su un altro tasto, ad esempio il tasto Ctrl destro o il tasto *F12*, che generalmente non si usano mai.

Fig. 2: **Opzioni della Virtual PC. Tastiera in differenti stati**

Questo per evitare un problema di Virtual PC con le tastiere europee. Se decidete di non cambiarlo e la tastiera iniziasse a fare le bizze, dovete premere il tasto Ctrl sinistro per sbloccarla. In pratica, usando Alt-Gr a volte nella VM rimane premuto il tasto Ctrl, rendendo impossibile la digitazione. Premendolo lo si sblocca.

SISTEMI OPERATIVI SUPPORTATI

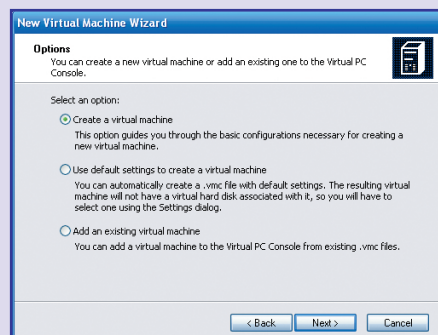
Per una lista aggiornata dei sistemi operativi funzionanti o meno sotto VPC si faccia riferimento a questo sito: http://vpc.visualwin.com. È molto comodo perché vengono anche indicate eventuali note per portare a termine l'installazione con successo. Si noti come se anche non supportato ufficialmente, è possibile installare Linux su VPC. Selezionare i modi video a 16bit con driver VESA, in quanto i modi video a 24 bit non sono supportati da Virtual PC.

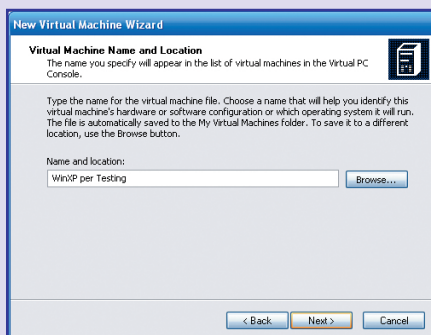
CREIAMO UNA NUOVA VM

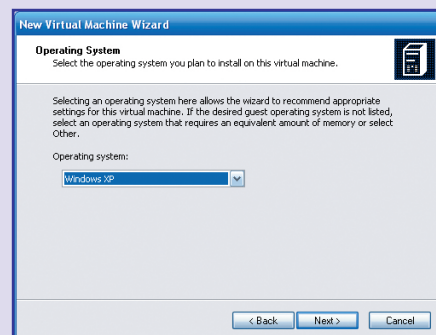
Ci sono vari modi per creare una nuova VM. Il modo più semplice è illustrato nel tutorial in basso. Una volta creata una nuova VM con il relativo harddisk, si può installare il sistema operativo ospite. Prima di tutto si deve avviare la macchina virtuale. Ci si rende subito conto che è identica ad un PC vero. Si parte con il BIOS che effettua la sequenza di boot. Ma l'hard disk è vuoto, quindi bisogna installare un nuovo sistema operativo. Per farlo si procede come per un PC vero. Si inserisce il CD di boot o il floppy di boot. Poi tramite il menu CD o il menu floppy si seleziona di "catturare" il dispositivo nella VM, per poterlo utilizzare. A questo punto si segue la sequenza di installazione del sistema operativo, con il riconoscimento dell'hardware e l'installazione dei vari componenti. Si possono montare anche file .iso al posto dei CD veri e propri, e si

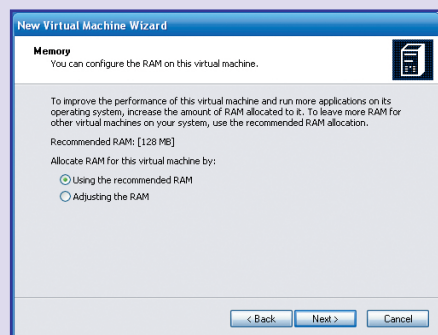
UNA MACCHINA VIRTUALE IN 6 PASSI

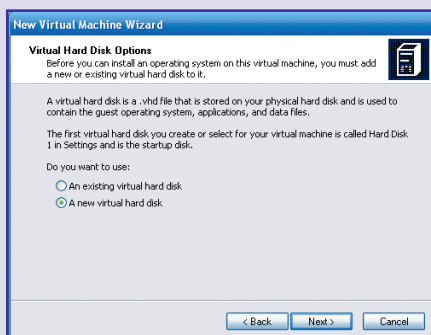
Con poche semplici operazioni creiamo una nuova macchina pronta ad ospitare un completo sistema operativo

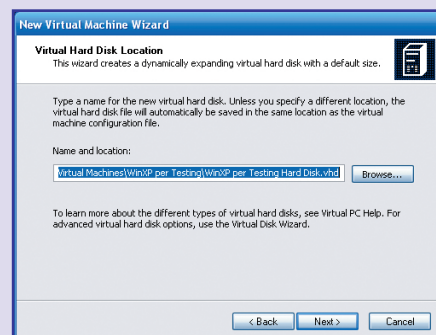
> CREIAMO UNA NUOVA VM

1 Per cominciare scegliamo di creare una nuova VM, completa di Hard Disk virtuale.

> DIAMOGLI UN NOME

2 Il nome che sceglieremo per la VM sarà anche il nome del file .vmc che verrà creato.

> SCEGLIAMO IL SISTEMA

3 Individuiamo quale sistema installare. A breve sceglieremo quanta Ram allocargli.

> QUANTA?

4 La quantità suggerita è sempre il minimo indispensabile. Se potete aumentatela.

> CREIAMO UN NUOVO HD

5 Per cominciare è meglio scegliere di creare un nuovo HD virtuale vuoto. Lo riempiremo in seguito.

> DOVE LO SALVIAMO?

6 Decidiamo il nome e la posizione per il file .vhd che conterrà l'HD virtuale. E siamo pronti.

possono usare file .vfd al posto dei floppy. Su questo sito è possibile trovare le immagini dei floppy per i sistemi operativi più diffusi: *http:// www. roudybob.net/?p=123*

INSTALLIAMO LE VM ADDITIONS

Le prestazioni e le funzionalità di una VM sono limitate se non vengono installate le VM Additions. Queste servono per abilitare alcune funzionalità, come ad esempio le risoluzioni avanzate dello schermo, l'integrazione del puntatore del mouse, la condivisione della clipboard fra sistema operativo ospite e ospitante, la condivisione dei dischi, la sincronizzazione dell'orologio, e altro. Le Additions al momento sono disponibili per Windows, MS-DOS e OS/2. Per installare le Additions, bisogna selezionare l'apposita voce dal menu Action. Se si aggiorna Virtual PC, ad esempio con un Service Pack, bisogna ricordarsi di aggiornare anche le Additions, per non avere problemi.

ALTRI PRODOTTI SIMILI

Microsoft vende anche un prodotto chiamato Virtual Server, che è ottimizzato per sistemi operativi server. Una serie di prodotti concorrenti sono prodotti da VMWare www.vmware.com e comprendono una versione Workstation e varie versioni server. Esistono poi prodotti OpenSource, come Xen e Bochs, ma non hanno ancora raggiunto la completa maturità.

PREPARIAMO L'AMBIENTE DI TEST

La nostra VM è pronta per essere utilizzata per i test, ma prima conviene fare qualche ottimizzazione all'ambiente per renderci la vita più facile. La prima cosa da fare è quella di abilitare i dischi di Undo, in modo da poter tornare alla situazione originale dopo aver effettuato il test. Per farlo bisogna selezionare la macchina virtuale, verificare che sia spenta (stopped) e premere il tasto Settings. Nella pagina Undo Disks è possibile abilitarli. Un'altra modifica che consiglio vivamente è quella di abilitare l'integrazione del Mouse e di condividere il disco del sistema operativo ospitante usando le Shared Folder. Io ad esempio condivido il disco C: del sistema ospitante con la lettera Z: nel sistema ospite. Le Shared Folder sono comode perché consentono

Fig. 3: *Opzioni della VM. Undo disk*

di condividere file senza dover mettere in piedi delle condivisioni di rete esplicite. Un altro modo di condividere file è di trascinarli direttamente con un drag&drop dal sistema ospitante a quello ospite. L'integrazione del Mouse è molto comoda, e permette di non dover premere il tasto Host per rilasciare il mouse, che altrimenti resterebbe catturato nella VM.

TESTIAMO IL SETUP DI UN PROGRAMMA

Vediamo ora come utilizzare in pratica una VM per testare il setup di un programma. In un ambiente normale dovremmo fare diverse prove: installazione, disinstallazione, reinstallazione, etc… Se troviamo un errore nella procedura di setup, dobbiamo formattare la macchina e reinstallarla, perché altrimenti le prove successive potrebbero essere falsate da eventuali componenti rimasti in sospeso. Con Virtual PC e i dischi Undo attivati, si può procedere in questo modo. Si lancia la VM, si fa l'installazione, se tutto è ok si testano gli altri scenari (disinstallazione, reinstallazione, etc…), mentre se qualcosa va storto, si può chiudere la finestra della VM. A questo punto una dialog ci chiede cosa vogliamo fare. Nella combo di scelta avete l'opportunità di eliminare tutte le modifiche fatte dall'ultimo salvataggio permanente dei dati. In questo modo ripartirete da una situazione pulita. Se invece

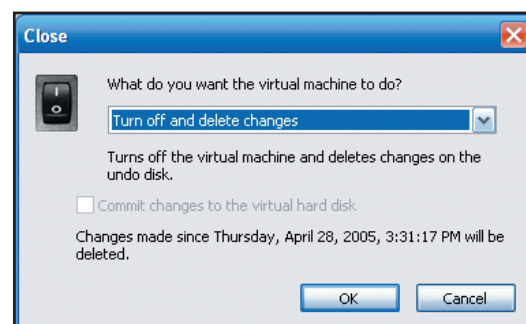



Fig. 4: *Spegnimento con Undo disk*

GLI STATI POSSIBILI

Le VM possono essere in stati differenti: accese, spente, sospese o salvate. Se una VM è accesa la sua finestra con il desktop dello schermo viene mostrata a video (si può anche andare in full screen), se la VM è sospesa la finestra è ancora mostrata a video, però ingrigita. La VM continua però ad occupare memoria. La sospensione (pausa) è comoda quando ci serve il processore sul SO ospitante per effettuare dei brevi lavori. Altrimenti conviene salvare la VM. Tutto il suo stato viene memorizzato su harddisk e la VM viene ibernata, permettendoci di ripartire esattamente dove l'avevamo lasciata. In questo modo la VM non occupa più memoria e la sua finestra a video viene chiusa.

volete rendere le modifiche permanenti, dovrete scegliere un'opzione che preveda il salvataggio dei dati, e selezionare "commit changes to the virtual harddisk" per renderle permanenti. Una volta rese permanenti non è più possibile tornare indietro. Vi consiglio di preparare un sistema operativo con tutto quello che serve (patch, service pack, applicazioni, etc..) e di rendere persistenti le modifiche. A quel punto lo si può usare come base per futuri test.

PREPARIAMO CONFIGURAZIONI MULTIPLE

Nel caso si debbano testare configurazioni multiple, conviene preparare la configurazione base e poi spegnere la VM. A questo punto si può prendere il file .vhd associato alla VM e copiarlo. Una volta copiato, si può creare un'altra VM dicendo di usare il nuovo file come HD virtuale, e automaticamente abbiamo un'altra VM pronta all'uso. Se le due VM vengono usate contemporaneamente sulla stessa rete, bisogna però cambiare il SID (l'identificatore) della macchina. Si veda il box "Ottimizzare la configurazione di macchine simili" per maggiori informazioni.

PERIFERICHE SUPPORTATE

Virtual PC supporta HD, CD, Floppy veri e virtuali, schede di rete, porte seriali e parallele vere (solo se sono sulla motherboard) e virtuali (scrivendo ad esempio su file di testo), tastiere e mouse USB. La scheda video emulata è una S3, quella audio è una SoundBlaster a 16bit.

Non sono supportate in questa versione periferiche USB, ad eccezione di tastiere e mouse, e degli HD/memory pen ma solo se riconosciute come dischi dal sistema operativo ospitante. Virtual PC non permette inoltre l'accesso ai dispositivi come schede ISA, PCI, PCMCIA, etc...

COLLEGAMENTO IN RETE PER TEST DI APPLICAZIONI CLIENT-SERVER

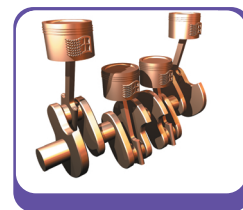
Molto spesso si ha la necessità di testare applicazioni client-server, ma non si dispone del numero sufficiente di macchine. Se si ha abbastanza RAM, è possibile effettuare il test in locale utilizzando due o più VM collegate in rete. È possibile configurare una VM in varie modalità:

- **Not connected** – questa è l'opzione più semplice. La VM è disconnessa da qualsiasi rete.
- **Local** – questa VM può vedere solo le altre VM impostate come local.
- **Shared networking** – con questa impostazione la scheda di rete viene condivisa con il sistema operativo ospitante, ed entrambe le macchine vengono viste in rete.

È anche possibile prendere una scheda di rete e dedicarla completamente ad una VM. Se non si hanno schede di rete installate nel PC, o se si volesse creare una rete di tipo Local, ma che include anche il SO ospitante, si potrebbe installare il Microsoft Loopback Adapter (è un adattatore di rete virtuale, installabile dal CD del sistema operativo) e utilizzarlo come scheda di rete virtuale.
Gli scenari che si aprono collegando in rete più VM sono molto interessanti: è possibile ad esempio avere una macchina server Linux o Windows e delle macchine client miste Windows /Linux in varie versioni per testare ad esempio un sito Web con vari browser. Il server può essere sempre attivo, mentre i client possono essere attivati e disattivati dalla sospensione a piacimento, per non occupare tutte le risorse contemporaneamente.

CONCLUSIONI

In questo articolo introduttivo si sono messi in evidenza i pregi di Virtual PC per il test del software. È possibile ad esempio installare versioni ancora in beta senza la paura di dover formattare la macchina se ci sono problemi di compatibilità o bachi.
Un ultimo consiglio: installate sempre l'ultimo Service Pack di Virtual PC, perché è molto più ottimizzato della versione base, soprattutto se usate Windows XP Service Pack 2.

Lorenzo Barbieri

NOTA

PROVARE VIRTUAL PC SENZA ACQUISTARLA

Sul sito Microsoft è disponibile la versione trial che funziona senza limiti per 45 giorni.

L'AUTORE

Lorenzo Barbieri è laureato in Ingegneria Informatica e lavora come Trainer e Consulente sulle tecnologie Microsoft. E' Microsoft Certified Trainer ed è certificato MCSD.NET e MCDBA su SQL Server 2000. Può essere contattato attraverso il suo blog http://blogs.ugidotnet .org/lbarbieri dove potrete trovare molte altre informazioni su Virtual PC e .NET

OTTIMIZZARE LA CONFIGURAZIONE DI MACCHINE SIMILI

**Copiare i file .vhd per clonare le macchine alla lunga provoca una notevole occupazione del disco. In questo articolo online viene spiegato come sfruttare una feature avanzata di Virtual PC, i dischi differenziali, che permettono di avere dischi base e dischi che contengono solo le modifiche effettuate.
A fronte di un disco base ci possono essere diversi dischi differenziali, e questo è molto comodo ad esempio se si deve testare un programma con versioni differenti di Office o del Service Pack installato.
Si installa il sistema operativo nel disco base, e poi si differenziano i dischi differenziali successivamente. Viene anche mostrato come cambiare il SID della VM per evitare conflitti in rete.
Per maggiori informazioni:** http://blogs.ugidotnet .org/lbarbieri/articles /VPC2004SP1_HDVirtuali Differenziali.aspx

SOFTWARE SUL CD

ANT
Il costruttore di applicazioni

Molti di voi sicuramente conosceranno l'utility make. Viene usata soprattutto in C/C++ per effettuare il “Build” di applicazioni che necessitano di molte dipendenze. Normalmente un makefile contiene una serie di macroistruzioni che servono a gestire la compilazione condizionale, a compilare i file nel giusto ordine oppure ancora ad ottimizzare le opzioni di compilazione sulla base del sistema operativo su cui l'applicazione dovrà girare. Ant offre l'analogo servizio ma per Java. Si tratta di un tool utile spesso usato anche in congiunzione ad applicazioni già esistenti, per ottimizzare l'uso di una libreria o di un'applicativo all'interno delle proprie strutture di programmazione.

Directory: /ant

APACHE
Il web server per eccellenza

Chi non conosce Apache? Si tratta di uno dei Web Server più usati al mondo. A contendergli lo scettro da qualche tempo c'è IIS. Ma Apache per grado di sicurezza assicurato, per numero di moduli disponibili, per quantità di documentazione resta ancora saldamente in testa.

Ovviamente, la sua leadership è ancora più salda in ambienti Unix, tuttavia trova sempre più posto anche in sistemi Microsoft.

Certo in ambiente Windows non riuscirete con Apache a fare girare la pagine Asp/.NET a meno di effettuare alcuni tuning, tuttavia Apache rimane il sistema di riferimento per un quantitativo straordinario di web application scritte soprattutto in PHP

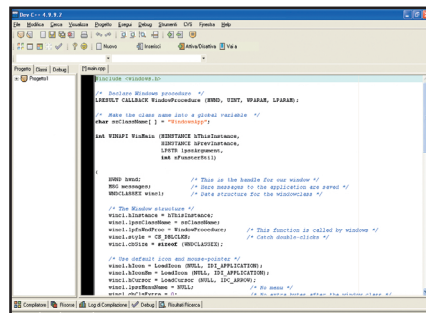
Directory: /apache

BAMBOO
Per utilizzare la “prevalenza” in .NET

Di prevalenze parliamo in questo numero di ioProgrammo. Scoprirete che si tratta di una tecnica del tutto innovativa che consente di trattare i database in maniera totalmente diversa da come eravate abituati a fare fino ad ora. In particolare, grazie alla prevalenza si riesce a trasferire il controllo delle interrogazioni dal file system alla memoria, operazione che velocizza e non di poco qualsiasi tipo di database. Ma c'è molto di più dietro al concetto di prevalenza, leggete il bell'articolo di Antonino Panella e lo scoprirete. Bamboo è la libreria che consente di applicare la tecnica ad applicazione .NET.

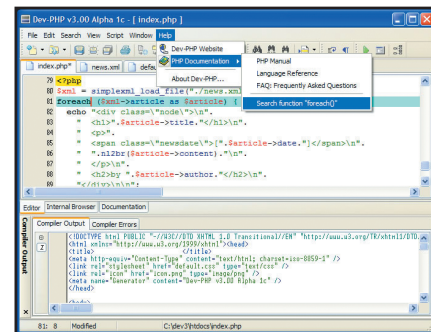
Directory: /bamboo

DEVCPP
Il più amato dai programmatori C++

C++ è ancora il linguaggio principe per i duri e puri della programmazione e per quanti vogliono un controllo assoluto sul comportamento di un qualunque applicativo. Per programmare in C++ non è necessario un IDE pesante o strumenti particolarmente complessi. Basta un compilatore e poco altro.

A testimoniare l'efficienza del linguaggio c'è Dev C++ che in pochi Kb di codice racchiude un IDE utilissimo per la programmazione C++ dotato di tute le funzionalità che possono aiutare il moderno programmatore nel corso dello sviluppo e tuttavia privo di tutti quei fronzoli che spesso rendono difficoltoso il concentrarsi esclusivamente sul codice. Per questi motivi Dev C++ si è conquistato nel tempo e a ragione il ruolo di leader negli IDE gratuiti per C++.

Directory: /devcpp

DEVPHP
L'IDE free per la programmazione PHP

La programmazione PHP necessita di un web server compatibile, del linguaggio e del solo NotePad per editare i file.

Certo se si ha a disposizione un comodo IDE dotato di Syntax Hilighting, di code complexion e altre funzionalità che rendono più immediato lo sviluppo del codice tutto cambia. DevPHP è un IDE leggero, completo, comodo e gratuito che vi mette in grado di scrivere abbastanza velocemente i vostri script PHP.

Sicuramente un prodotto da tenere sempre installato per coloro che utilizzano PHP come linguaggio principe per la scrittura delle proprie applicazioni.

Directory: /devphp

Macromedia Dreamweaver 8

La nuova versione dell'IDE per la creazione di siti WEB

Molte le novità in arrivo con la nuova versione dell'editor più utilizzato dai designer/programmatori del Web.
Prima di tutto una migliore gestione dei fogli di stile così come dell'ormai onnipresente XML grazie a un nuovo strumento visuale. Sono stati inseriti alcuni nuovi strumenti come ad esempio il pannello di rendering che rende immediata la visualizzazione di tipi di media eterogenei.
Infine Dreamweaver 8 si integra perfettamente con tutti gli altri componenti della suite Macromedia Studio 8, adesso il passaggio da un'applicazione a un'altra nei vari formati è veramente trasparente.

Directory: /dreamweaver

ECLIPSE

L'ambiente tuttofare

Eclipse nasce da un'intuizione straordinaria, ovvero quello di essere un ambiente estensibile per la programmazione.
Non per la programmazione con un solo linguaggio ma in generale con ogni linguaggio. Installando il plugin giusto vi troverete sotto mano sempre un IDE aggiornato per lo sviluppo con il vostro linguaggio preferito. Inoltre tutte le funzionalità necessaria ad un buon programmatore per sviluppare rapidamente il proprio codice sono garantite dalla base comune dell'ambiente, il che rende Eclipse uno strumento assolutamente straordinario.

Aggiungete tutto ciò che si tratta di un IDE gratuito e multipiattaforma e scoprirete perché rapidamente è diventato il preferito dai programmatori.

Directory: /eclipse

IRRLICHT

L'engine 3D per la creazione di VideoGiochi

In questo numero di ioProgrammo presentiamo il *"terrain mapping"*, una tecnica assolutamente straordinaria che consente di disegnare ambienti aperti, come distese d'erba, montagne, terreni in modo assolutamente realistico. Per questo articolo, il nostro Alfredo Marroccelli che da tempo ci accompagna nel mondo del VideoGaming, ha scelto di usare l'engine Irrlicht che per caratteristiche e funzioni messe a disposizione rappresenta uno dei motori più interessanti attualmente sulla scena.
Un motore che non deve mancare nella dotazione di ogni programmatore di videogiochi che si rispetti.

Directory: /irlicht

J2SE

L'sdk essenziale per la programmazione Java

Se siete dei programmatori Java o aspirate a diventarlo non potete mancare di installre il J2SE di Sun, al cui interno è contenuto il compilatore

Macromedia Flash 8

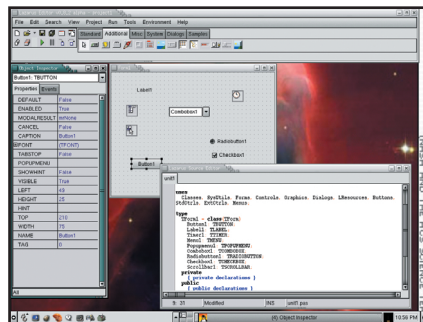
La nuova versione del leader nel settore multimedia

Le novità più evidenti in Macromedia Flash 8, oltre alla consueta migliorata integrazione con gli altri componenti della suite, riguardano in larga parte l'uso dei formati di compressione video.
È stato introdotto un nuovo formato denominato On2 VP6 che riduce di molto l'ingombro del filmato finale in fase di pubblicazione sul web. Tutte le altre novità presenti soltanto nella versione professional sono relative alla gestione del codec. Ulteriori miglioramenti sono stati apportati alla gestione del canale Alpha e all'integrazione di nuovi filtri molto simili a quelli usati da Photoshop.

Directory: /flash

nonché tutte le librerie essenziali per potere programmare in Java. Quella che vi presentiamo è l'ultima release aggiornata del compilatore che copre molti bachi di programmazione e si presenta leggermente più veloce.
Directory: /j2se

LAZARUS
Il clone gratuito di Delphi

I programmatori più anziani ricorderanno Delphi come il primo ambiente realmente RAD ad essere stato immesso sul mercato. È grazie a Delphi se oggi ragioniamo in termini di componenti, ed è grazie a Delphi che è nato il concetto di Rapid Application Development. Non mancherebbe davvero niente a questo ambiente per farne il leader indiscusso del mercato, se non fosse che un prezzo particolarmente elevato e qualche strategia non troppo azzeccata lo hanno relegato ad essere uno degli ambienti preferiti in applicazioni di tipo Business, ma non il più diffuso in ambienti Home/Office.

A modificare radicalmente la situazione ci pensa Lazarus che, se da un lato, non contiene tutte le raffinate soluzioni proposte da Delphi, dall'altra parte, ha dalla sua il fatto di essere completamente Free e di rappresentare un clone praticamente perfetto della versione base di Delphi . Si tratta di un ambiente veramente interessante, da provare, sia in ambienti di produzione, sia se volete assaggiare la potenza dell'Object Pascal prima di passare a Delphi.
Directory: /lazarus

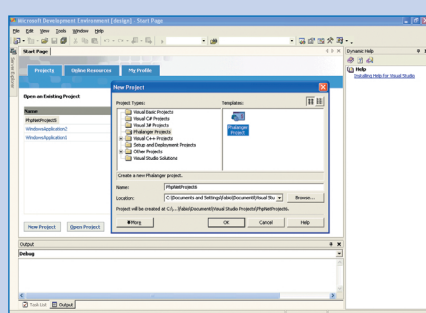
MYSQL
Il DB più usato sul web

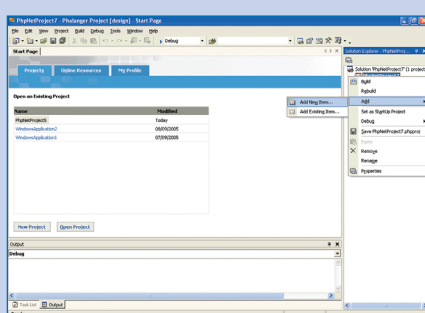
In congiunzione a PHP, MySQL fa girare buona parte del Web. Non c'è applicazione PHP che non utilizzi in qualche modo MySQL per gestire e recuperare i dati. Una così enorme dif-

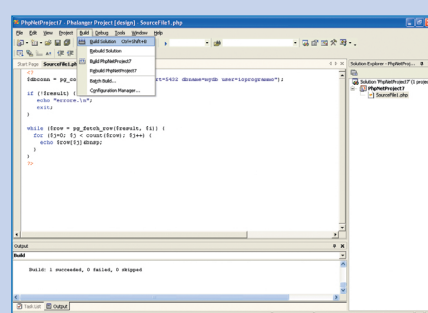
PHALANGER
Per eseguire applicazioni PHP compilandole in .NET

Phalanger è un tool decisamente innovativo. Il suo funzionamento è banale da un punto di vista fisico, più complesso da un punto di vista logico.
Da un lato consente di sviluppare applicazioni PHP tramite Visual Studio.NET.
Queste applicazioni possono essere ricompilate sotto forma di eseguibile. Dal punto di vista del web Phalanger non ha bisogno del classico interprete PHP per funzionare, viceversa utilizza un modulo .NET che ricompila gli script e affida l'esecuzione al .NET Framework. In pratica si tratta di PHP per .NET. Esattamente come esiste C#,VB.NET con cui potete creare le vostre pagine .ASPX adesso potete immaginare di usare PHP .NET con tutti i vantaggi della tecnologia in questione in ambienti Microsoft.
Il processo di installazione è molto semplice, ma per motivi di licenza non abbiamo potuto distribuire tutto il necessario sul CD in allegato e ci siamo dovuti limitare al solo file *setup.exe* di configurazione. Lanciandolo vi verranno richiesti dei prerequisites.
Potete scaricare il tutto gratuitamente da www.php-compiler.net. Per effettuare il deploy di un'applicazione sul web, è necessario creare un file *web.config*, dopo avere installato Phalanger troverete molti esempi di *web.config* pronti all'uso.
Noi vi lasciamo con un tutorial che mostra come usare Phalanger da Visual Studio.NET 2003
Directory: /phalanger

PHALANGER IN 3 PASSI
Ecco come costruire un'applicazione PHP con Visual Studio.net

› UN NUOVO PROGETTO

1 Cliccate su *file/new/Project* e selezionate *Phalanger* dal menu ad albero sulla sinistra e ancora una volta l'icona Phalanger nella finestra dei progetti. Preoccupatevi anche di dare un nome e un percorso corretto alla soluzione.

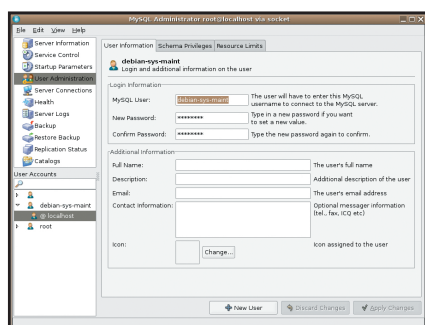
› UN NUOVO ITEM

2 Dal *solution explorer* cliccate con il tasto destro del mouse sull'icona che rappresenta il progetto e dal menu a tendina scegliete *"Add Item"*. Nella successiva dialog box scegliete *"Source File"*. Date un nome al nuovo file e continuate.

› COMPILATE ED ESEGUITE

3 Scrivete il codice del vostro progetto PHP all'interno dell'editor. Notate che Visual Studio si adatterà al codice PHP utilizzando la *Sintax Highliting*. Quando sarete sicuri del risultato, cliccate su *Build/Build Solution* e costruite il vostro *.exe.*

fusione si uò giustificare solo con la bontà del prodotto. Infatti MySQL è un database estremamente leggero, gratuito, facile da usare e configurare. Si certo, manca ancora di qualcuna delle raffinatezze che fanno degli altri database dei mostri sacri per la programmazione in ambiente business, tuttavia anche questo gap va lentamente colmandosi. Per questi motivi MySQL sta rapidamente scalando posizioni anche nella programmazione di applicazioni standalone, anche se il suo ambito di riferimento primario rimane il web.
Directory: /mysql

MYSQL ADMINISTRATOR

Per amministrare MySQL in modo efficace

Fino a qualche tempo fa l'unico modo di amministrare MySQL era attraverso la linea di comando, oppure tramite applicazioni prodotte da terze parti, non sempre adeguate agli standard qualitativi di questo database.

Attualmente le cose sono radicalmente cambiate. MySQL Administrator è un tool squisitamente grafico, prodotto da AB, la stessa software house produttrice di MySQL che consente di amministrare a colpi di click il vostro database preferito. Se è vero che l'avvento delle interfacce grafica ha rivoluzionato il mondo della programmazione, l'arrivo di questo genere di tool sicuramente aiuta di molto il sistemista nella gestione quotidiana del database.
Directory: /mysqladmin

MY SQLMIGRATION TOOLKIT

Per migrare da un database proprietario a MySQL

A quanto pare MySQL vuole fare le cose in grande, così ecco arrivare un wizard completamente grafico per migrare i dati da db come Oracle e Access a MySQL. Si tratta di una bella comodità e di un invito esplicito a provare MySQL anche in ambiti dove fino ad ora non aveva trovato terreno fertile. Se è logico infatti migrare da Access a MySQL sulla base di considerazioni legate alla multiutenza, alla velocità e a tutti i vantaggi di questa migrazione, bisogna essere sicuri di offrire un prodotto competitivo per chiedere di migrare da un mostro sacro come Oracle a MySQL.
Directory: /mysqlmt

MYSQL QUERY BROWSER

Per eseguire query e gestire un db MySQL in modo semplice

Esattamente come nel caso precedente, MySQL Query Browser appartiene a quella schiera di interfacce grafiche che fanno da ponte tra l'utente e MySQL. In questo caso si tratta di uno strumento per gestire le tabelle come le colonne di uno o più database ed eventualmente per facilitare l'immissione di Query SQL.
Directory: /mysqlbrowser

PHP

Il linguaggio più usato del WEB

Ormai è noto, come PHP sia uno dei linguaggi più diffusi su internet. La sua curva di apprendimento bassissima, la completezza del linguaggio, e con la versione 5 anche il notevole supporto agli oggetti ne fanno il punto di riferimento per un incredibile numero di programmatori. La sua natura free ne ha consentito una distribuzione larghissima e una disponibilità di documentazione senza pari. Si tratta senza dubbio di un must per chi programma applicazioni Internet.
Directory: /php

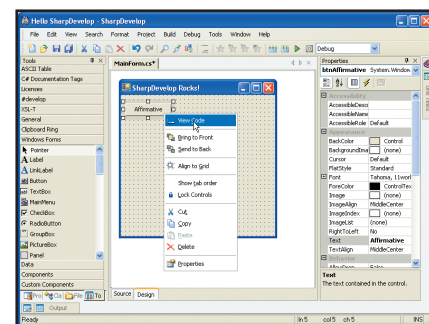
PREVAYLER

Il framework per gestire la prevalenza in Java

Di *prevalenza* ne scrive, in questo stesso numero di ioProgrammo, Antonino Panella. Si tratta di una tecnica di cui sentiremo parlare sempre più spesso, in quanto realmente innovativa e capace di produrre una svolta nel difficile mondo dei database. Con la prevalenza, molte operazioni comunemente svolte tramite accesso al disco, vengono trasferite alla memoria, ottenendo un miglioramento delle prestazioni eccezionali. A qualcuno potrebbe sembrare un'assurdità, ma se avrete la pazienza di leggere l'articolo presentato in questo stesso numero, scoprirete che la tecnologia è davvero interessante e ormai matura per essere immessa sul mercato.
Directory: /prevayler

SHARPDEVELOP

Lo strumento per programmare in C# senza Visual Studio

C# è il linguaggio proposto da Microsoft in .NET rivolto agli sviluppatori professionali. Si tratta di un ottimo linguaggio che si muove in parte sulla falsa riga dei concetti già espressi da Java. Una pecca importante sta nei costi dell'ambiente. Visual Studio risulta almeno nelle versioni attuali sufficientemente impegnativo dal punto di vista dei costi tanto da non garantire un largo accesso a tutti.

A risolvere il problema ci pensa Sharp Develop, IDE completamente gratuito, che mantiene tutte le caratteristiche più importanti di un ambiente RAD, pur non avendo la complessità e ovviamente il costo di Visual Studio.
Directory: /sharpdevelop

TOMCAT

L'application Server per JSP e le Servlet

Se siete dei programmatori JSP avete senza dubbio bisogno di Tomcat per testare e utilizzare le vostre applicazioni. È senza dubbio un *server wen* ovvero può funzionare da server Web ma è soprattutto un Application Server che vi consente di utilizzare la tecnica delle servlet o delle JSP per creare applicazioni Web con una logica business e con il linguaggio Java a fare da asse portante.
Directory: /tomcat

Programmare sudoku

Il passatempo più in voga del momento arriva dal Giappone e si chiama sudoku. La programmazione svela alcuni interessanti aspetti del gioco e ci intriga quanto, o addirittura di più, del gioco stesso

L'ultima passione degli enigmisti di tutto il mondo si chiama Sudoku. Il rompicapo intrattiene le menti dei giocatori più brillanti ormai da qualche tempo e può a ragione, considerarsi ormai un cult. Un passatempo che ha intrattenuto molti. Il coinvolgimento è stato a livello planetario, visto che si è giocato in Europa come in America come in Asia da dove proviene. A dire il vero pare che il gioco non sia del tutto nuovo e che fosse già conosciuto prima ancora che si diffondesse come un virus. Sicuramente un gioco molto simile, ma possiamo dire anche uguale: "number place" era conosciuto negli Stati Uniti negli anni settanta. Ma rimandiamo una mini ricerca storica ai prossimi appuntamenti. Riveliamo, per i pochi che ne sono all'oscuro, le regole del gioco e poi incrociamole con le risorse messe a disposizione dalla programmazione per computer.

LE REGOLE DEL GIOCO

La versione standard che è la più diffusa prevede un quadrato, ossia una griglia di 81 caselle, 9 per riga e 9 per colonna. Bisogna tenere conto dei 9 quadratini dimensione tre righe e tre colonne che sono naturalmente collocati all'interno della griglia principale, così come mostrato in **Figura 1**. A seconda della difficoltà verranno inizialmente immessi all'interno del quadrato dei numeri. È questo il compito del compositore dello schema di sudoku. L'obiettivo è quello di incasellare i restanti numeri, i primi nove naturali (da 1 a 9), in tutte le caselle della griglia in modo che vengano rispettate tre semplici quando fondamentali regole:

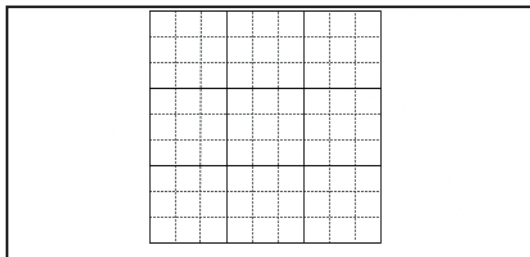



Fig. 1: *La griglia di partenza del sudoku*

1. Ogni riga deve presentare tutti i numeri compresi nell'intervallo [1..9] senza ripetizioni;
2. Ogni colonna deve presentare tutti i numeri compresi nell'intervallo [1..9] senza ripetizioni;
3. Ogni quadrato, dei nove a disposizione, deve presentare tutti i numeri compresi nell'intervallo [1..9], senza ripetizioni.

Il rompicapo ricorda vagamente i quadrati magici ma come si può osservare rispetto ad essi ha sostanziali differenze.

Fig. 2: *La griglia risultato del sudoku. Output del programma*

PROGRAMMAZIONE: PRIMO PASSO

La programmazione può incontrare il gioco a diversi livelli. Può verificare se una soluzione sia esatta, andando a verificare per ogni riga, ogni colonna e ogni quadrato che non si presentino ripetizioni di numeri. È questo un compito niente affatto banale e

che può essere di aiuto al giocatore che dopo analisi superficiale potrebbe pensare erroneamente di aver risolto il suo schema. È ciò che faremo tra breve. Un secondo compito è quello di generare soluzioni a partire da composizioni iniziali. In tal caso il programma si sostituisce al giocatore. Si tratta di un utile esercizio che fa capire in profondità le regole da adottare in una soluzione per così dire manuale. Infine, un programma può proporre una composizione da sottoporre ai giocatore. In questo caso si sostituisce al compositore. È questo il compito più delicato. Ma passiamo al nostro primo compito la verifica di una soluzione. Lo facciamo in C++.

QUADRATI MAGICI

Il compositore dello schema, di solito un enigmista proporrà una serie di numeri che inizialmente costituiranno la griglia. A seconda del numero e di come sono disposti varierà la difficoltà. Normalmente si evitano schemi che abbiano molti numeri altrimenti la soluzione degenererebbe in una mera scrittura di numeri. Invece, è necessario che si facciano delle scelte seguendo ragionamenti logici; per questo i numeri che sono presenti inizialmente non sono molti, comunque sempre più di 20. In rari casi 18 o 19. Ovviamente, esistono delle regole che il solutore dovrà seguire per arrivare a completare l'intero quadro.

Usiamo una matrice *mat* di dimensione nove per nove di short (numeri interi piccoli). Ricordo che gli indici di tale struttura andranno da un minimo di zero a un massimo di otto. Per il controllo dei numeri per riga, colonna e quadrato useremo un array di booleani di dimensione 10, ma che per comodità sarà usato nell'intervallo di indici da 1 a 10. La variabile logica *check* servirà come controllo globale di tutta la griglia. Ecco la dichiarazione delle strutture usate nel programma

```
bool check, checkp[10];

short mat[9][9];
```

Di seguito sono riportate tutte le funzioni sviluppate per la realizzazione del compito di controllo della griglia.

```
void visual()
{ short i,j;
  char stop;
   for (i=0; i<9; i++ )
   {  cout<<'\n';
      if (i%3==0) cout<<'\n';
      for (j=0; j<9; j++)
      {      if (j%3==0) cout<<'\t';
             cout<<mat[i][j]<<' ';}  };
   cout<<"\n\n"; };
void azzera()
{   short i;
    for(i=1;i<10;i++)
    checkp[i]=false;  };
bool controlla(char dove,short k)
{  short j, jdiv, kdiv;
   bool ch=true;
   azzera();
   if (dove=='r')
  // Controlla riga k
  for (j=0; j<9; j++)
      checkp[mat[k][j]]=true;
   else if (dove=='c')
      // Controlla colonna k
      for (j=0; j<9; j++)
          checkp[mat[j][k]]=true;
      else
      // Controlla quadrato k
      for (j=0; j<9; j++)
      {   jdiv=j/3;
          kdiv=k/3;
          checkp[mat[jdiv+kdiv*3][((j%3)+
                              (k%3)*3)]]=true; };
    // da checkp si ottiene la singola variabile di uscita
                                            (return)
    for (j=1; j<10; j++)
      {  cout<<checkp[j]<<" x ";
         ch = (checkp[j] && ch); };
       cout<<'\n';
    return ch; };
bool verifica()
{   bool ch=true;
    short z;
    for (z=0; z<9 ; z++)
      {  cout<<ch<<'\n';
         ch = (ch && controlla('r',z) && controlla('c',z)
                          && controlla('q',z)); } ;
    return ch;
};
```

Commentiamole con ordine in sequenza. La funzione *visual()* ha il semplice scopo di riportare a video la griglia completa, il suo output è riportato in **Figura 2**. La funzione *azzera()* configura inizialmente l'array *checkp* ponendo i nove elementi di interesse a *false*. La funzione *controlla(char, short)* verifica che l'elemento specificato dal primo parametro (*r: riga, c: colonna* e *q quadrato*) sia corretto, ossia contenga i nove distinti numeri. Per farlo ogni numero che incontra nell'insieme in esame cambia lo stato del corrispondente casella del vettore *checkp* a *true*. Così se alla fine della scansione di nove elementi tutti e nove sono *true* allora quell'insieme (riga, colonna o quadrato) è corretto. Mentre, è abbastanza banale la parte riferita alle righe e alle colonne, ottenuta con un semplice indirizzamento indiretto su *checkp* mediante il valore corrente della matrice; merita un commento il controllo dei quadrati. I quadrati sono numerati come espresso in

Fig. 3: ***Schematizzazione della griglia. Numeri blu quadrati; verdi righe e rossi colonne***

Figura 3 dai numeri blu. Il parametro *k* passato alla funzione *controlla()* indica appunto il quadrato che stiamo controllando. Inoltre, con il ciclo presente scandisce la variabile *j* da 0 a 8, tutti gli elementi di un insieme, nel caso specifico un quadrato. In sostanza dalla coppia di valori *k* e *j* dobbiamo ottenere gli indici assoluti della griglia. Per farlo si sono usate le operazioni elementari su numeri interi di resto e quoziente, indicate con gli operatori C++ di % e /. Questa seconda diventa quoziente se applicata come nel nostro caso a numeri interi. Dopo attenta analisi ho dedotto che la riga si ottiene da operazioni di quoziente mentre la colonna di resto. Ad esempio per la riga il quoziente su *k* mi riporta su una delle tre "fasce" costituite da gruppi di tre quadrati disposti in orizzontale, e il quoziente su *j* mi colloca all'interno della fascia. Consiglio un'attenta analisi dell'espressione prodotta. La ritengo un risultato stilisticamente notevole che esprime il massimo della sintesi.
La funzione *verifica()* predispone un ciclo, e con la chiamata, in un'espressione logica della funzione controlla vengono ispezionate: la riga, la colonna e il quadrato *z*. Se tutte restituiscono vero *(true)* l'espressione totale che è un *and* logico tra tutte, sarà ancora vera. Basterà trovare un solo insieme che indica falso e tutta l'espressione diventerà falsa. Nel programma principale infatti viene testato il risultato di *verifica()* che potrà essere vero o falso.

```
int main()
{   char stop;
    carica();
    visual();
    if (verifica()) cout<<"Griglia corretta";
    else cout<<"Griglia errata";
    cin>>stop;
};
```

La funzione *carica()* semplicemente inserisce i valori da controllare nella matrice *mat.* La variabile stop è usata per bloccare il risultato e poterlo così visionare. Di fatto viene invocata la sola funzione *verifica()* che a sua volta richiama le altre routine descritte, ovvero *controlla()* e *azzera().*

UNA SOLUZIONE AUTOMATICA

Adesso proveremo a trovare una soluzione. Ci renderemo subito conto che il compito è una dura sfida per il programmatore. Adotteremo un "backtracking intelligente". Cosa intendo dire con ciò? Che faremo uso della conosciuta tecnica di esplorazione esaustiva delle soluzioni il backtracking a cui assoceremo delle scelte che ottimizzeranno la produzione della soluzione. Ricordo che il backtracking è una tecnica che procede per tentativi. Nel caso del sudoku il linea di massima, al generico passo di assegnazione di un nuovo numero in una nuova casella, funziona così:

- Produce una soluzione è verifica se è possibile adottarla, se è così la inserisce altrimenti ne produce una nuova;
- Se nessuna delle soluzioni prodotte da esito positivo, ovvero nessuna può essere adottata per i vincoli dettati dalle regole del gioco, allora si fa un passo indietro (back) e si somministra lo stesso procedimento alla soluzione precedente. Ovviamente tenendo conto delle soluzioni già esaminate.

Iterando il processo si perviene in conclusione ad una soluzione verificabile dalla funzione prima sviluppata. Per realizzare il metodo è quindi necessario uno *stack* (pila) che ci consenta di fare una *push* (inserimento nella pila) di una soluzione quando questa è corretta, e una *pop* (estrazione) per tornare un passo indietro al fine di valutare altre soluzioni. Nel progettare la struttura dati adeguata è indispensabile prendere in considerazione alcune esigenze. La più rilevante è trovare un modo per verificare in modo rapido una soluzione. Bisogna in altri termini, una volta prodotta una soluzione parziale, ossia un numero da piazzare in una casella, controllare che non sia già presente nella stessa riga, colonna o quadrato. Tale compito potrebbe essere anche sviluppato semplicemente controllando i tre insiemi in questione direttamente nella matrice principale mat. Ma per chiarezza di codice e soprattutto per efficienza, è opportuno definire una nuova matrice di booleani che per ogni insieme e per ogni collocazione ci indichi se la soluzione è già presente. Una leggera ridondanza che velocizza l'algoritmo. Chiameremo tale array a tre dimensioni *rcq*. Il primo indice segnala il tipo di insieme, secondo la semplice corrispondenza *1: riga; 2; colonna; 3: quadrato.* Il secondo equivale all'insieme si sta esaminando (esempio la terza colonna, o il quarto quadrato). Il terzo indice è l'effettivo numero che si inserisce. Con il vettore *bloccate* possiamo tenere traccia delle caselle che non si possono toccare. Questo serve quando bisogna trovare una soluzione a fronte di una composizione enigmistica, ossia quando ci sono già dei nu-

SUDOKU E QUADRATI MAGICI

I quadrati magici sono griglie quadrate in cui vanno incasellati dei numeri. Fin qui uguale al sudoku. In cui per ogni riga, colonna e per le due diagonali la somma deve convergere ad un unico numero. I numeri da inserire sono anche nel caso dei quadrati magici appartenenti all'intervallo [1..n] dove n è m al quadrato, con m la dimensione del quadrato, ossia il numero di righe o colonne.
In figura è riportato un quadrato magico di ordine 3.

6	7	2
1	5	9
8	3	4

Quadrato magico di dimensione 3"

meri all'interno della griglia che ovviamente non si possono cambiare. Nel nostro primo esempio, ad ogni modo, anche se prevediamo tale possibilità, risolveremo il gioco a fronte di una griglia pulita, senza alcun numero inserito. La variabile *prog* è molto importante e indica il numero di soluzione che si sta tentando di risolvere. Mentre, la variabile *max* esprime le soluzioni da trovare, esattamente 81 se la griglia è inizialmente pulita. Infine, pila è lo stack costituito da elementi base disco. Tale record contiene tutto ciò che serve per tener traccia di una soluzione parziale. Il progressivo (la soluzione *dprog*), il numero che si è tentato di inserire *dneo*, e tutte le soluzioni che sono esplorate, nonché il numero di soluzioni esplorate *(nesp)*. Le soluzioni esplorate sono memorizzate mediante un vettore di booleani chiamato *espl* dove l'iesimo elemento è vero se la soluzione è esplorata, falso altrimenti. Ecco quindi i nuovi tipi e le nuove variabili globali che vanno aggiunte al codice precedentemente mostrato.

```
short mat[9][9];
bool rcq[4][9][10];
// matrice che indica la presenza di un numero in
// uno dei tre set in esame quadrato, riga e
// colonna; rende più rapida la consultazione
// e l'inserimento dei numeri
short max; //numero di caselle libere inizialmente;
short prog; //numero di caselle effettivamente
                                        collocate
bool bloccate[81];
// Dichiarazioni inerenti la pila
struct disco
{  short dprog, dneo; // numero casella;
                                 soluzione attuale
   bool espl[10]; // soluzioni esplorate
   short nespl; //numero di soluzioni esplorate };
disco pila[81];
disco sol;
short lev; // livello della pila;
```

NOTA

TIPO BOOL

È un tipo logico che può assumere uno tra due valori, true e false. Viene quindi usato anche all'interno di espressioni logiche, come per il programma prodotto per la verifica di sudoku.

Le funzioni di base e di servizio sono routine che servono tutte per il buon funzionamento del metodo cruciale ovvero *risolvi()*. Le prime si occupano di gestire la pila mediante un array. A tale scopo sono usate le funzioni standard di gestione dello *stack*.
Le seconde effettuano dei controlli per verificare se una soluzione è stata esplorata o meno *(esplorata)* e se è corretta o errata *(errata)*. Per farlo basterà controllare in modo opportuno le strutture dati costruite. In particolare, una soluzione è esplorata se il valore booleano del vettore *espl* (che tiene traccia dei numeri) della soluzione corrente è vero, altrimenti se falso vuol dire che quella soluzione non è mai stata tentata. Una soluzione è, invece, corretta se il numero non è presente in nessuno dei tre insiemi sensibili: riga, colonna e quadrato. La verifica si effettua semplicemente controllando i tre valori corrispondenti della matrice *rcq*. Infine, con la routine satura abbiamo modo di sapere se sono state osservate tutte le possibilità in una generica soluzione parziale. Quando il numero di esplorazioni *(nespl)* raggiunge 9 vuol dire che siamo giunti alla saturazione di una insieme di tentativi. Come vedremo quando ciò accade bisognerà fare un passo indietro *(back)* e tentare nuove vie, usando appunto la *pop*.

```
// Funzioni inerenti la pila;
bool is_empty()
{   return lev==0; };
bool is_full()
{  return lev==81; };
disco pop()
{  return pila[lev--]; };
disco top()
{  return pila[lev]; };
void push(disco d)
{   pila[++lev]=d; } ;
// Funzioni di supporto a componi()
bool esplorata(short pneo, disco psol)
{    return psol.espl[pneo]; };
bool errata(short pneo, short pprog)
{   short eriga, ecolo, eqriga, eqcolo;
    eriga=pprog/9;
    ecolo=pprog%9;
    eqriga=eriga/3;
    eqcolo=ecolo/3;
    return ( (rcq[1][eriga][pneo]) || (rcq[2][ecolo]
      [pneo]) || (rcq[3][eqriga*3+eqcolo][pneo]) ); };
bool satura(disco psol)
{     return psol.nespl==9;
};
```

La funzione principale è riportata di seguito. Nella prima fase si attua una semplice inizializzazione delle variabili e delle strutture precedentemente descritte. Poi con il ciclo di *while* controllato dalla variabile prog si innesca la parte più importante mediante la quale si risolve il sudoku. Eventualmente la casella sia bloccata si procede all'automatica esplorazione di nuove soluzioni con l'incremento di *prog*. Il nuovo numero da collocare si genererà casualmente, si tratta di *neo*. La randomizzazione assicura una più efficiente ricerca delle soluzioni, anche a tale proposito si parlava di backtracking intelligente. Qualora sia errato si generano in sequenza nuove soluzioni. Quando si esce dal ciclo bisogna osservare se la soluzione corrente è corretta *(not errata)*, tale da consentire la *push* o satura tale da richiedere la *pop*. Se non ci troviamo in nessuna delle due situazioni si procede con una nuova iterazione del ciclo alla ricerca di nuove soluzioni. Vorrei anche in questo frammento di codice far notare alcuni particolari. Ad esempio, la trasformazione del numero di soluzione *prog* in elementi utili

all'implementazione. Abbiamo già visto come si trasforma in riga e colonna. Ma è altrettanto interessante la trasformazione nel quadrato di competenza. Dalla divisione intera *(div)* tra la riga con tre e la colonna con tre e la successiva composizione dell'espressione *qriga*3+ qcolo*, una vera rarità. Ne consiglio l'analisi. Alcuni cout sono stati riportati dal sottoscritto per tenere traccia delle operazioni svolte.

```
void risolvi()
{  short neo; //nuovo numero che si tenta di collocare
   short bl;
   char st;
   bool vis;
   short j,i,s,riga,colo,qriga,qcolo; // variabili di servizio
    // configura bloccate
    for (bl=0;bl<=81;bl++)
    bloccate[bl]=false;
    // confifura rcq
    for (s=1; s<=3; s++)
    for (i=0; i<9; i++)
    for (j=1; j<=9; j++)
    rcq[s][i][j]=false;
    // predisponi soluzione iniziale
    disco sol;
    // genera la prima soluzione
    sol.dprog=prog;
    for (j=1; j<10; j++) sol.espl[j]=false;
    sol.nespl=0;
    while (prog<=80)
    {  while (bloccate[prog])
        prog++;
        neo=1+rand()%9;
        while ( errata(neo,prog) && !satura(sol) )
        {  if (!esplorata(neo,sol)){  sol.dneo=neo;
              sol.espl[neo]=true;
              sol.nespl++; }   ;
           neo=rand()%9+1; };
        if ((!errata(neo,prog)) && (!esplorata(neo,sol))){
              sol.dneo=neo;
              sol.espl[neo]=true;
              sol.nespl++;
              push(sol);
              riga=prog/9;
              colo=prog%9;
              qriga=riga/3;
              qcolo=colo/3;
              //Assegno soluzione
              mat[riga][colo]=neo;
              cout<<"[ "<<riga<<" , "<<colo<<" ] -->
                                  "<<neo<<"\n";
              rcq[1][riga][neo]=true;
              rcq[2][colo][neo]=true;
              rcq[3][qriga*3+qcolo][neo]=true;
              prog++;
              cout<<prog<<"\n";
              // genera una nuova soluzione
              sol.dprog=prog;
              for (j=1; j<10; j++) sol.espl[j]=false;
              sol.nespl=0; }
        else if (satura(sol))
            if (!is_empty())
            { cout<<"POP";
              sol=pop();
            riga=sol.dprog/9;
            colo=sol.dprog%9;
            qriga=riga/3;
            qcolo=colo/3;
            //riazzero la matrice;
            mat[riga][colo]=0;
            // ripristina rcq
            rcq[1][riga][sol.dneo]=false;
            rcq[2][colo][sol.dneo]=false;
            rcq[3][qriga*3+qcolo][sol.dneo]=false;
            prog=sol.dprog;
            cout<<prog<<"\n";} }
}
```

Ecco un possibile modo per richiamare tutte le funzioni sviluppate, che come ripeto si occupano di trovare una soluzione al sudoku e di verificarne la correttezza.

```
int main()
{   char stop;
    prog=0;
    max=81;
    lev=-1;
    carica();
    risolvi();
    visual();
    if (verifica()) cout<<"\tGriglia corretta";
    else cout<<"\tGriglia errata";
    cin>>stop;
};
```

```
6 9 8   5 2 4   1 3 7
4 1 7   6 8 3   2 9 5
3 2 5   1 7 9   6 4 8

8 6 2   3 9 7   5 1 4
7 3 9   4 1 5   8 2 6
1 5 4   8 6 2   3 7 9

2 8 3   7 4 6   9 5 1
9 7 1   2 5 8   4 6 3
5 4 6   9 3 1   7 8 2
```

***Fig. 4:* Output di sudoku.cpp**

In **Figura 4** è riportato parte dell'output del programma. In particolare, la parte più importante che mostra la griglia corretta, frutto dell'algoritmo sviluppato.

CONCLUSIONI

In problemi come questi chiaramente gioca un ruolo fondamentale l'efficienza del codice prodotto. A tale proposito la soluzione proposta sicuramente si presta a ulteriori miglioramenti. Oppure, si possono tentare strade completamente diverse per affrontare la questione. Sono molto curioso su cosa Voi lettori ne pensiate, e in questo senso spero che si apra una discussione che porti a dipanare tutti i nodi della questione. Il punto di riferimento è come sempre il forum di ioProgrammo *http://forumioprogrammo.it.*

Fabio Grimaldi

ON LINE

JAVA PORTAL

Ricchissimo di spunti, informazioni, esempi di codice. L'obiettivo di JavaPortal non è solo quello di proporsi al pubblico come un centro di raccolta di informazioni, quanto quello di rivitalizzare una comunità di programmatori Java abbastanza vasta da essere in grado di realizzare applicazioni significative per l'intero panorama informatico.

http://www.javaportal.it

DEVLEAP

Devleap propone una serie di articoli di "qualità" relativi alla programmazione in ambienti Microsoft. Il 23 e 24 Novembre si terrà a Milano la Devleap Conference 2005 che vedrà la partecipazione di un nucleo molto ristretto di programmatori che si cimenteranno nell'apprendimento di tecniche piuttosto esclusive.

http://www.devleap.com/

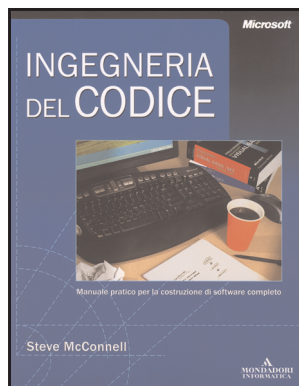
DOT NET EXPERTS

Viene proposta una suddivisione in gruppi di discussione molto simile a quella dei newsgroup, all'interno di ciascun gruppo si trovano argomenti piuttosto interessanti. In quasi ogni gruppo inoltre si trovano consigli, opinioni e commenti di esperti del settore, che altrimenti si potrebbero trovare solo in conference tecniche, oltre che naturalmente in ioProgrammo.

http://www.dotnetexperts.com/

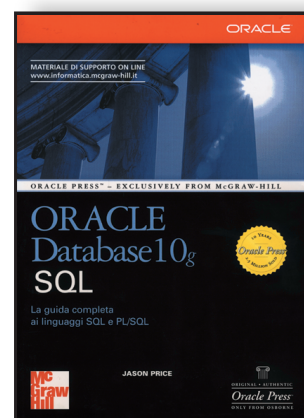
Biblioteca

INGEGNERIA DEL CODICE

Il sottotitolo di questo libro recita: "Manuale pratico per la costruzione di software completo". Mai un sottotitolo è stato più generico eppure più azzeccato. Il libro si presenta come una guida semplice alla costruzione di software robusto, flessibile e sicuro. L'idea è semplice, fornire al lettore un'insieme di conoscenze che lo mettano di scrivere programmi migliori e in meno tempo. Al contempo si tenta di colmare il divario fra la ricerca universitaria, spesso fumosa e inconsistente, e le necessità delle aziende che invece vogliono vedere in quella che è pura teoria un vantaggio economico o di impresa per il loro Business. Se per un tema talmente astratto potrebbero sembrare sufficienti poche pagine contenenti le solite scarse informazioni sull'argomento, sorprenderà invece che il manuale si snoda in ben 914 pagine che partono dalla basi fino a coprire argomenti quali l'autodocumentazione del codice oppure l'incidenza delle dimensioni del programma nella sua costruzione. Si tratta di un libro affascinante quanto utile per coloro che affrontano la programmazione con la professionalità che necessita a questo mondo.

Difficoltà: Alta • **Autore:** Steve McDonnell • **Editore:** Mondadori • **ISBN:** 88-04-54034-6 • **Anno di pubblicazione:** 2005 • **Lingua:** Italiana • **Pagine:** 984 • **Prezzo:** € 80,00

ORACLE DATABASE 10G

Oracle è ancora uno dei database più usati al mondo. Offre un numero di funzionalità realmente elevato. Ciascuna di queste funzionalità è stata pensata secondo standard di qualità elevatissimi tesi a soddisfare le reali esigenze dei programmatori. Il libro di Jason Price è una guida esaustiva a SQL e PL/SQL in relazione alle estensioni apportate dai progettisti di Oracle al linguaggio. Il libro è utilizzabile sia dagli utenti base che dai professionisti, si parte dal primo capitolo: "Che cos'è un database relazionale" fino ad arrivare a rispondere negli ultimi a capitoli a esigenze quali "L'esecuzione di codice Sql in Java". La struttura del libro è piuttosto fluida, i cntenuti chiari, il numero di esempi elevati, così che apprendere in modo rapido le nozioni fondamentali per lavorare con Oracle non sarà difficile. La conoscenza di base di Oracle é sicuramente un elemento di conoscenza importante per ogni programmatore.

Difficoltà: media • **Autore:** Steve McConnell • **Editore:** McGrawHill • **ISBN:** 88-386-4414-4 • **Anno di pubblicazione:** 2004 • **Lingua:** Italiana • **Pagine:** 614 • **Prezzo:** € 51,00

ROBOT BUILDING

Questo libro appartiene alla schiera di quelli che stuzzicano decisamente la fantasia. Da un lato l'approccio sembra essere serioso, si parla di elettronica come di programmazione dei microcontrollori, d'altra parte l'idea di creare piccoli robottini che vanno in giro per casa a compiere questa o quella funzione è decisamente stuzzicante. Il libro è scritto in lingua inglese, ma lo stile è colloquiale quanto basta per non risultare estremamente difficile da comprendere o da tradurre. David Cook parte dalle basi dell'assemblaggio per poi affrontare argomenti più complessi quale l'uso di sensori di movimento, di temperatura oppure l'uso degli infrarossi per l'individuazione degli ostacoli. Come già detto si tratta di un libro decisamente interessante, da leggere se siete il tipo di persona a cui piace lasciarsi stupire da tutto ciò che è tecnologicamente innovativo. Da un punto di vista pratico il libro é un'ottima occasione per imparare a conoscere la programmazione dei robot; un campo in rapida espansione.

Difficoltà: media • **Autore:** David Cook • **Editore:** Apress • **ISBN:** 1-59059-373-1 • **Anno di pubblicazione:** 2004 • **Lingua:** inglese • **Pagine:** 422 • **Prezzo:** € 34,00